매니코어 시스템에서 네트워크 버퍼 할당 정책에 따른 성능 분석

엄준용O, 조중연, 진현욱
건국대학교 컴퓨터공학부
{oursoon2, dynamicj, jinh}@konkuk.ac.kr

Performance Analysis of Network Buffer Allocation Policy over Manycore Systems

Junyong UhmO, Joong-Yeon Cho, Hyun-Wook Jin
Department of Computer Science & Engineering, Konkuk University

요 약

NUMA 기반의 매니코어 시스템에서 구조적 특징에 의해 프로세스가 수행되는 코어 또는 버퍼가 할당되는 메모리 위치에 따라 네트워크 I/O 성능이 달라질 수 있다. 네트워크 I/O 장치는 수신된 패킷을 처리하기 위해 커널 수준과 응용 수준에서 상당한 수의 메모리 접근이 발생한다. 따라서 네트워크 I/O에 집중적인 프로세스가 사용하는 버퍼 할당 정책에 따라 네트워크 I/O 성능에 영향을 미칠 수 있다. 본 논문에서는 응용 수준의 버퍼 할당 정책에 따른 네트워크 I/O 성능을 실험하고 버퍼 할당 정책이 네트워크 성능에 미치는 영향에 대해 분석한다.

1. 서 론

멀티코어 시스템의 구조적 특징에서 기인된 코어 친화도 개념은 I/O 성능 향상을 비롯해 하드웨어 자원을 효율적으로 사용할 수 있는 가능성을 제공한다. 특히 네트워크 I/O 성능 관점에서 코어 친화도에 따라 네트워크 I/O 성능이 달라질 수 있음은 기존의 연구들을 통해 보여 지고 있다[1,2]. 코어 친화도와 함께 NUMA 기반의 멀티코어 시스템에서 프로세서와 메모리 간의 물리적인 거리에 의해 야기되는 문제들을 완화시키기 위한 연구들도 진행되었다[3,4].

네트워크 I/O 장치는 수신된 패킷을 처리하기 위해 DMA 과정을 거쳐 패킷을 메모리에 복사한다. 고속 네트워크 I/O 장치의 경우 짧은 시간동안 많은 양의 패킷이 수신되며 커널 수준의 TCP/IP 프로세싱 및 데이터 복사가 집중적으로 발생되기 때문에 메모리 I/O에 집중적일 수 있다. 따라서 NUMA 구조를 사용하는 시스템에서 커널 수준 또는 응용 수준의 버퍼 할당 정책에 따라 네트워크 I/O 성능에 영향을 미칠 수 있다.

본 논문에서는 NUMA 구조를 사용하는 매니코어 시스템에서 응용 수준의 네트워크 버퍼 할당 정책에 따른 네트워크 I/O 성능을 측정한다. 성능 측정과 더불어 프로세서의 PMC(Performance Measurement Counter) 레지스터를 이용해 캐시 히트율과 지역/원격 메모리 접근 정보를 수집한다. 실험 결과를 기반으로 네트워크 버퍼 할당 정책이 네트워크 I/O 성능에 미치는 영향을 분석한다.

본 논문은 다음과 같이 구성되어 있다. 본 서론에 이어 2장에서는 NUMA 구조의 메모리 특성과 관련 연구에 대해 설명하고, 3장에서는 네트워크 버퍼 할당 정책에 따른 네트워크 I/O 성능을 실험을 통해 보여준다. 마지막 4장은 본 논문의 결론 및 향후 계획을 기술한다.

이 논문은 2015년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.B0101-15-0644, 매니코어 기반 초고성능 스케일러블 OS 기초연구)

2. 연구 배경

2.1 NUMA 구조에 따른 메모리 특성

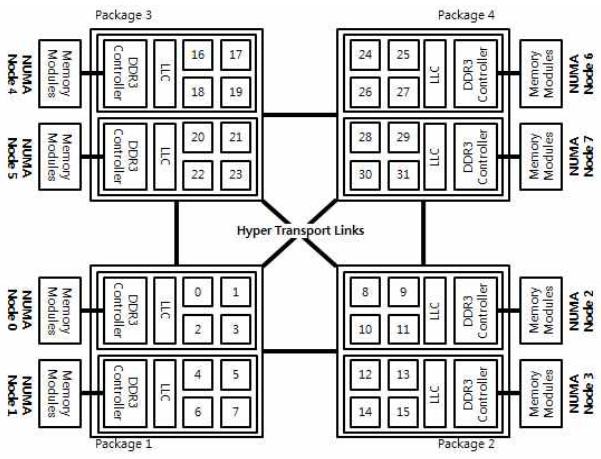

그림 1 4-Way AMD Magny-Cours 시스템 구조

본 절에서는 NUMA 구조에 따른 메모리 접근 속도의 특성에 대해 간략히 설명하고 네트워크 I/O 관점에서 발생할 수 있는 문제점에 대해 기술한다. NUMA 구조의 가장 큰 특징은 프로세서 패키지마다 메모리 노드를 보유한다는 점이다. 그림 1은 4-Way AMD Magny-Cours 프로세서[5]

로 구성된 시스템 구조를 나타낸다. 그림 1에서 볼 수 있듯이 Magny-Cours 프로세서의 경우 하나의 프로세서 패키지에 두 개의 다이가 조합되어 있는 형태로 하나의 프로세서 패키지는 두 개의 NUMA 노드로 구성되어 있다. 그림 1과 같이 네 개의 프로세서 패키지를 사용하는 시스템은 총 8개의 NUMA 노드로 구성된다. 일반적으로 AMD 프로세서의 경우 지역 메모리 접근은 약 100 Cycle이 소모되며 원격 메모리 접근은 수백 Cycle이 소모된다. 만약 코어가 데이터에 접근하기 위해 지속적으로 원격 메모리에 접근한다면 상대적으로 많은 Cycle을 소비하기 때문에 네트워크 I/O 성능은 물론 자원 사용 효율을 감소시킬 수 있다.

2.2 관련 연구

본 절에서는 관련 연구들에 대해 기술한다. NUMA 구조로 이루어진 멀티코어 시스템에서 메모리 위치에 따른 영향에 대한 연구는 주로 스케줄링 관점에서 진행되고 있다. 프로세서 부하와 메모리 부하를 계층적으로 고려하여 프로세스를 스케줄링하기 위한 연구[3]와 응용 수준 스케줄러를 제안한 연구[4]들이 진행되었다. 또한 NUMA 구조를 고려하여 메모리 자원 경쟁을 줄이기 위한 연구도 진행되었다[5]. 리눅스 운영체제 역시 운영체제 수준의 지원을 계속해 나아가고 있다.

하지만 기존의 연구들은 네트워크 I/O 집중적인 프로세스에서 사용하는 버퍼에 대한 고려가 미흡하다. 실험 역시 프로세서 집중적 또는 메모리 집중적인 마이크로벤치마크를 이용하여 시스템 부하 관점에서 논의하고 있어 네트워크 I/O 성능에 대한 부분은 포함되지 않는다. 본 논문에서는 네트워크 I/O에 집중적인 프로세스를 이용해 네트워크 버퍼 할당 정책에 따른 네트워크 I/O 성능을 실험하고 결과를 분석한다.

3. 네트워크 버퍼 할당 정책에 따른 영향 분석

본 장에서는 네트워크 버퍼 할당 정책에 따라 변화하는 네트워크 I/O 성능을 실험하고 결과에 대해 논의한다. 실험에는 그림 1과 같은 구조로 이루어진 NUMA 구조 기반의 매니코어 시스템을 사용하였다. 이 시스템은 네 개의 AMD Opteron 2.0GHz 옥타코어 프로세서로 구성되어 있다. 데이터 전송을 위한 클라이언트 머신은 하나의 Intel 3.4GHz 쿼드코어 프로세서로 구성되어 있다. 운영체제는 각각 리눅스(커널 버전 3.10.0-123)를 사용하였으며 네트워크 인터페이스 카드는 Chelsio사의 10Gbps 이더넷 컨트롤러를 사용하였다.

3.1 네트워크 버퍼 할당 정책

본 절에서는 네트워크 버퍼 할당 정책에 대해 기술한다. 본 논문에서는 네트워크 버퍼 할당 정책을 캐시 효과 관점과 네트워크 버퍼 위치 관점으로 분류 하였다.

네트워크 버퍼를 반복해서 재사용하는 경우 네트워크 버퍼 영역이 최하위 레벨 캐시에 존재할 가능성이 높아진다. 하지만 네트워크 버퍼를 새롭게 할당하는 경우 버퍼 할당 및 사용을 위해 메모리에 직접 접근한다. 따라서 캐시 효과에 의해 네트워크 I/O 성능이 달라질 수 있다. 본 논문에서는 캐시 효과 관점의 네트워크 I/O 성능에 대해 실험하기 위해 ttcp 마이크로벤치마크[7] 알고리즘을 이용해 두 가지 경우에 대해 각각 구현하고 실험에 사용하였다.

캐시 효과와 더불어 네트워크 버퍼의 위치에 따라 네트워크 I/O 성능이 달라질 수 있다. 2.1절에 기술한 바와 같이 프로세서와 메모리의 물리적인 거리에 따라 접근에 필요한 Cycle 수가 크게 차이나기 때문이다. 또한 AMD Magny-Cours 프로세서의 경우 프로세서 패키지간의 연결을 위한 HT(HyperTransport)의 속도 차이가 있기 때문이다. 본 논문에서는 네트워크 버퍼의 위치를 지역 노드, 인접 노드, 원격 노드와 같이 세 가지로 분류한다. 지역 노드는 응용 프로세스가 수행되는 코어의 NUMA 노드를 의미하며, 인접 노드는 같은 프로세서 패키지에 위치한 NUMA 노드(그림 1의 Package 1의 NUMA 노드 0과 1)를 의미한다. 원격 노드는 완전히 다른 프로세서 패키지에 위치한 NUMA 노드(그림 1의 NUMA 노드 0과 2)를 의미한다.

네트워크 버퍼 할당 정책과 함께 네트워크 I/O 성능에 영향을 줄 수 있는 요소들은 다음과 같이 설정하였다. 프로세스 친화도와 인터럽트 친화도는 가장 높은 네트워크 I/O 성능에 도달할 수 있도록 최하위 레벨 캐시를 공유하는 두 개의 코어에 분배하였다[8,9]. 또한 커널에 의해 프로세스 또는 네트워크 버퍼 위치가 마이그레이션 되지 않도록 리눅스 운영체제에서 제공되는 Auto-NUMA Balancing은 비활성화하였다. 실험을 진행하는 동안 Oproflie을 사용해 PMC 레지스터 정보를 기록하였다.

3.2 실험 결과 및 성능 영향 분석

본 절에서는 실험 결과에 대해서 기술하고 네트워크 버퍼 할당 정책이 네트워크 I/O 성능에 미치는 영향에 대해서 논의한다. 그림 2는 네트워크 버퍼 할당 정책에 따른 네트워크 I/O 성능 측정 결과를 보여준다. 가로축은 네트워크 버퍼의 위치를 의미하고 세로축은 네트워크 대역폭을 나타낸다.

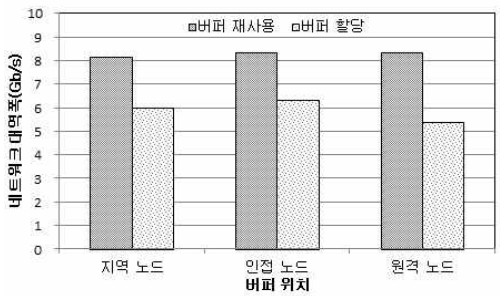

그림 2 버퍼 할당 정책에 따른 네트워크 대역폭

그림 2에서 볼 수 있듯이 네트워크 버퍼를 재사용하는 경우 평균 8.1Gbps의 대역폭에 도달한 반면 네트워크 버퍼를 새롭게 할당한 경우 평균 5.8Gbps의 대역폭에 도달한 것을 확인할 수 있다. 또한 네트워크 버퍼를 재

사용하는 경우 네트워크 버퍼의 위치에 따른 성능 영향이 보이지 않지만 네트워크 버퍼를 새롭게 할당한 경우 네트워크 버퍼 위치에 따라 네트워크 대역폭이 달라지는 것을 확인할 수 있다. 네트워크 버퍼 재사용 여부에 따라 네트워크 I/O 성능이 달라지는 원인은 캐시 효과에 의한 것으로 판단된다. 그림 3은 네트워크 버퍼 할당 정책에 따른 최하위 레벨 캐시 히트율을 나타낸다.

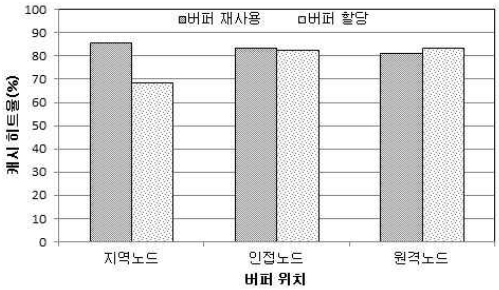

그림 3 버퍼 할당 정책에 따른 캐시 히트율

그림 3에서 볼 수 있듯이 네트워크 버퍼를 재사용 한 경우 버퍼의 위치와 관계없이 80% 이상의 높은 히트율을 보였다. 또한 네트워크 버퍼의 위치가 지역 노드일 때 캐시 히트율의 차이가 10% 이상이기 때문에 네트워크 I/O 성능에 영향을 미친 것으로 판단된다. 하지만 네트워크 버퍼를 새롭게 할당하고 버퍼의 위치가 인접 노드 또는 원격 노드일 때 80% 이상의 높은 히트율에 도달하고 있지만 네트워크 I/O 성능은 낮아졌음을 확인할 수 있다. 캐시 효과는 높아졌지만 원격 메모리 접근에 따라 시스템 버스 오버헤드가 증가했기 때문인 것으로 예상된다. 그림 4는 네트워크 버퍼 할당 정책에 따른 지역/원격 메모리 접근 횟수를 나타낸다.

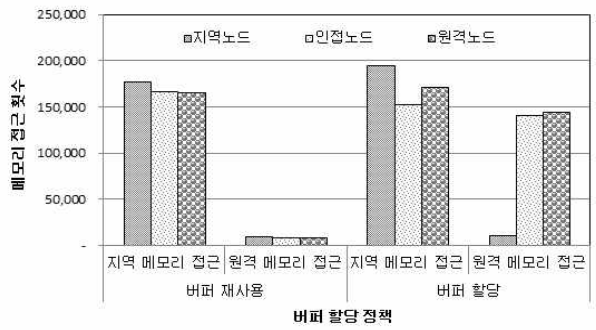

그림 4 버퍼 할당 정책에 따른 메모리 접근 횟수

그림 4에서 볼 수 있듯이 네트워크 버퍼를 재사용 한 경우 네트워크 버퍼의 위치와 관계없이 대부분 지역 메모리 접근으로 이루어진 것을 확인할 수 있다. 원격 메모리 접근이 없는 이유는 네트워크 버퍼 영역이 캐시되어 있기 때문에 실제로 메모리 접근을 하지 않은 것으로 판단된다. 하지만 네트워크 버퍼를 새롭게 할당한 경우 버퍼의 위치에 따라 원격 메모리 접근이 증가한 것을 확인할 수 있다. 네트워크 버퍼를 새롭게 할당하고 네트워크 버퍼 위치를 원격 노드에 위치시킨 경우 인접 노드와 메모리 접근 수 에는 큰 차이가 없지만 네트워크 대역폭은 15% 가량 차이가 나는 것을 확인할 수 있다. 3.1절에서 기술한 바와 같이 프로세서 연결을 위한 HT의 속도 차이에 의한 것으로 판단된다. 따라서 네트워크 I/O 성능을 향상시키기 위해서는 네트워크 버퍼의 캐시 효과를 최적화해야 하며 시스템 부하에 따라 캐시 효과를 최적화할 수 없다면 버퍼를 지역 노드에 할당해 시스템 버스 오버헤드를 최소화해야 한다.

4. 결론 및 향후 계획

본 논문은 NUMA 구조를 사용하는 매니코어 시스템에서 네트워크 버퍼 할당 정책에 따른 네트워크 I/O 성능에 대해 실험하였다. 실험 결과를 이용해 분석한 결과 네트워크 I/O 성능은 네트워크 버퍼의 캐시 효과와 네트워크 버퍼의 위치에 따라 달라질 수 있음을 보였다.

향후 계획으로는 커널 수준 버퍼 할당 정책에 따른 네트워크 I/O 성능에 대해 실험할 예정이다.

참고 문헌

[1] G. Narayanaswamy, P. Balaji, and W. Feng, “An analysis of 10-gigabit ethernet protocol stacks in multicore environments,” In Proc. of HotI, 2007.

[2] A. Foong, J. Fung, D. Newell, A. Lopez-Estrada, S. Abraham, and P. Irelan, “Architectural characterization of processor affinity in network processing,” In Proc. of ISPASS, 2005.

[3] D. Kang, H. Park, and J. Choi, "A Novel Memory-Aware CPU Allocation Policy for Multicore NUMA Architecture," In Proc. of RACS, 2010.

[4] S. Blagodurov and A. Fedorova, "User-level scheduling on numa multicore systems under linux," In Proc. of Linux Symposium, 2011.

[5] AMD, “Introduction to Magny-Cours,” http://developer.amd.com/resources/documentation-articles/articles-whitepapers/introduction-to-magny-cours/, 2010.

[6] S. Blagodurov, S. Zhuravlev, A. Fedorova, and A. Kamali, "A case for NUMA-aware contention management on multicore systems," In Proc. of ACM PACT, 2010.

[7] USNA, “TTCP: A test of TCP and UDP performance,” 1984.

[8] J.-Y. Cho, H.-W. Jin, M. Lee, and K. Schwan, " Dynamic core affinity for high-performance file upload on Hadoop Distributed File System," Parallel Computing, Volume 40, Issue 10, pp 722-737, 2014

[9] N. Hanford, V. Ahuja, M. Balman, M. K. Farrens, D. Ghosal, E. Pouyoul, and B. Tierney, "Characterizing the impact of end-system affinities on the end-to-end performance of high-speed flows, " In Proc. of ACM NDM, 2013.