멀티 코어 시스템에서 통신 프로세스의 스케줄링에 따른 성능 분석

장혜천, 진현욱
건국대학교 컴퓨터공학부
e-mail:{comfact, jinh}@konkuk.ac.kr

Impact of Process Scheduling on Network Performance over Multi-Core Systems

Hye-Churn Jang, Hyun-Wook Jin
Department of Computer Science and Engineering, Konkuk University

요 약

현재 멀티 코어 프로세서는 많은 서버에 적용되어 사용되고 있으며, 향후에는 하나의 프로세서 패키지에 포함될 코어의 개수는 계속해서 증가할 것이다. 그러나 현재 운영체제들은 멀티 코어 시스템을 멀티 프로세서 환경과 거의 동일하게 다루고 있으며 아직 멀티 코어 특성을 고려한 성능 최적화 시도는 미흡한 상태이다. 본 논문은 SMP와 NUMA 구조의 멀티 코어 프로세서 환경에서 통신 프로세스와 네트워크 인터럽트의 프로세서 친화도를 변화시키며 네트워크 처리율과 코어의 유휴 자원 양을 정량적으로 분석한다. 측정 결과 프로세서 친화도에 따라 통신 처리율은 크게 변하지 않지만 프로세서 자원의 요구량에는 크게 영향을 주는 것을 보인다. 또한 이러한 프로세서 자원의 영향은 멀티 코어 프로세서의 캐쉬 공유 구조 및 메모리 분산 구조와 밀접한 관계를 갖고 있음을 밝힌다.

1. 서 론

프로세서의 발열과 전력 문제를 해결하기 위해서 등장한 멀티 코어 프로세서는 현재 많은 서버에 적용되어 사용되고 있다. 향후에는 회로 집적 기술이 지속적으로 향상됨에 따라 하나의 프로세서 패키지에 포함될 코어의 개수는 계속해서 증가할 것으로 전망되며 그 활용 역시 크게 증가할 것이다. 그러나 현재 운영체제들은 멀티 코어 시스템을 멀티 프로세서 환경과 거의 동일하게 다루고 있으며 아직 멀티 코어 특성을 고려한 성능 최적화 시도는 미흡한 상태이다.

서버 시스템에서 서비스 제공을 위해서 수행되는 많은 응용 프로세스들은 통신을 수행한다. 서버 시스템들은 일반적으로 1Gbps 대역폭을 제공하는 기가비트 이더넷에서 10Gbps의 대역폭을 제공하는 10 기가비트 이더넷 [1], InfiniBand [2], Myrinet [3]까지 다양한 네트워크 연결을 사용하고 있다. 현재의 멀티 코어 프로세서들은 이들 연결 네트워크의 대역폭을 충분히 활용 가능할 정도의 계산 능력을 갖고 있다 [4].

이와 같이 멀티 코어 프로세서는 운영체제의 큰 수정 없이도 통신 성능과 관련된 많은 문제점을 해결하고 있는 것으로 보인다. 하지만 최근 진행된 연구들은 멀티 코어 구조를 고려한 성능 최적화의 필요성을 지적하고 있으며 [4, 5, 6, 7], 아직도 다음과 같은 질문에 대해서는 충분한 답을 얻지 못하고 있다: i) 코어 개수 증가에 비례하여 시스템의 유휴 자원이 증가하는가? ii) 프로세스 스케줄링이 통신 성능에 어떠한 영향을 미칠 수 있는가? iii) 통신 성능과 관련하여 멀티 코어 프로세서의 구조에 따른 고려사항이 있는가?

본 논문에서는 이들 질문에 대한 답을 Intel의 SMP 멀티코어 프로세서[8]와 AMD의 NUMA 멀티코어 프로세서 [9] 환경에서 정량적으로 측정된 결과와 함께 제시한다. 측정 결과는 통신 프로세스의 스케줄링이 코어의 유휴 자원 양에 밀접한 영향을 미치는 것을 보인다. 이와 같은 분석 결과는 멀티 코어의 특성을 고려한 프로세스 스케줄러를 구현하기 위해서 필수적인 데이터로서 활용될 수 있다.

본 논문은 다음과 같이 구성되어 있다. 서론에 이어서 2장에서는 통신 프로세스의 스케줄링에 따른 통신 처리율과 CPU 활용도를 측정한다. 그리고 3장에서는 측정 결과를 바탕으로 그 활용 방법에 대해서 논한다. 마지막으로 4장에서 본 논문의 결론을 내린다.

2. 성능 측정 방법론 및 결과

프로세서 친화도는 특정 처리를 가급적 같은 프로세서가 계속 담당하도록 함으로써 캐쉬 효과 등을 통해서 성

본 논문은 2008년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원(KRF-2008-331-D00450)과 2008년 한국전자통신연구원의 위탁과제 지원(7010-2008-0038)으로 연구되었음.

능 향상이 가능토록 한다. 통신과 관련된 프로세서 친화도는 다음과 같이 두 가지로 나눠볼 수 있다: i) 통신을 수행하는 응용 프로세스의 프로세서 친화도, ii) 네트워크 인터페이스 카드에서 발생하는 인터럽트의 프로세서 친화도. 본 논문에서는 리눅스 기반의 Intel SMP 멀티코어 프로세서와 AMD NUMA 멀티코어 프로세서 환경에서 이 두 가지 프로세서 친화도를 변경시키며 수신을 수행하는 동안의 네트워크 처리율(Mbps)과 코어의 유휴 자원 양(%)을 측정한다. 프로세스의 프로세서 친화도는 sched_setaffinity() 시스템 호출을 사용하여 설정하였으며, 인터럽트의 프로세서 친화도는 smp_affinty 파일을 수정하여 설정했다.

측정을 위한 서버 시스템은 다음과 같이 구성되었다. SMP 구조의 서버는 두 개의 Intel Quad-Core Xeon 5355 (Woodcrest) 프로세서와 4GB의 메모리를 장착하고 있다. NUMA 구조의 서버는 두 개의 AMD Quad-Core Opteron 2356 (Barcelona) 프로세서와 4GB의 메모리를 장착하고 있다. 따라서 두 서버 노드 모두 총 여덟 개의 코어를 갖고 있다. 그리고 두 노드 모두에 Silicom PXG4 기가비트 이더넷 네트워크 카드를 장착했다. 운영체제는 리눅스 커널버전 2.6.18을 설치하였다.

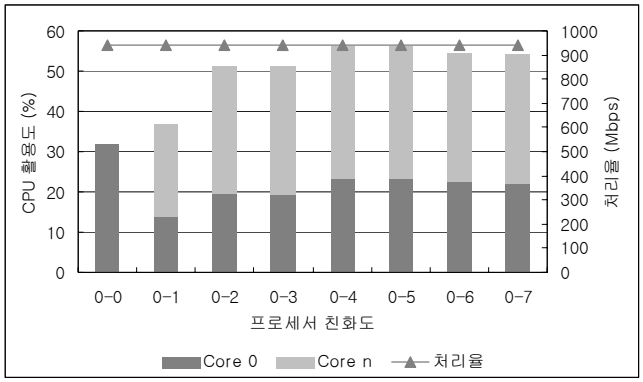

그림 1 SMP 멀티코어 구조에서 CPU 활용도와 처리율

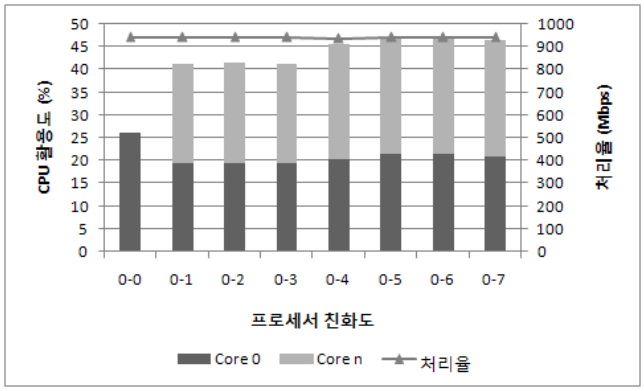

그림 2 NUMA 멀티코어 구조에서 CPU 활용도와 처리율

본 논문에서는 프로세서 친화도 설정을 <m-n>의 형식으로 표현한다. 여기서 m은 네트워크 인터럽트의 프로세서 친화도가 설정된 코어의 번호를 의미하며, n은 통신 응용 프로세스의 프로세서 친화도가 설정된 코어의 번호를 의미한다. 그림 1, 2의 x-축에서 볼 수 있는 바와 같이 측정에 사용된 프로세서 친화도는 <0-n>의 형태를 갖는다. 즉, 인터럽트의 프로세서 친화도는 항상 0번 코어에 할당하며, 통신 프로세스의 친화도는 0번에서 7번까지의 여덟 개 코어로 변화시키며 성능을 측정하였다.

그림 1은 SMP 멀티코어 구조에서 CPU 활용도와 통신 처리율 측정 결과를 보여주고 있다. 그림에서 볼 수 있는 바와 같이 프로세서 친화도에 따른 통신 처리율은 크게 영향을 받지 않는 것을 알 수 있다. 하지만 CPU 활용도는 상대적으로 크게 영향을 받고 있다. <0-0>의 프로세서 친화도가 가장 적은 CPU 자원을 요구하고 있다. 이것은 네트워크 데이터 수신을 위한 모든 처리가 0번 코어에서 수행되기 때문에 캐쉬 효과 등에 의한 것으로 분석된다. 반면 <0-1>의 경우는 <0-0>보다는 많은 CPU 자원을 요구하지만 다른 경우 보다는 낮은 CPU 자원을 요구하고 있다. 이것은 1번 코어가 0번 코어와 L2 캐쉬를 공유하기 때문에 다른 코어에 비해서 좋은 캐쉬 성능을 보이는 것에 기인한다. 하지만 캐쉬를 공유하지 않는 다른 코어들은 <0-0>의 경우보다 최대 27%의 CPU 자원을 더 요구하고 있다.

그림 2는 NUMA 멀티코어 구조에서 CPU 활용도와 통신 처리율 측정 결과를 보여주고 있다. SMP 멀티코어의 경우와 같이 NUMA 환경에서도 <0-0>의 프로세서 친화도 설정이 가장 적은 CPU 자원을 사용하고 있다. 그런데 Intel 멀티코어 프로세서와는 다르게 AMD 멀티코어 프로세서의 모든 코어는 각각 독립적인 L1, L2 캐쉬를 갖고 있다. 그리고 L3 캐쉬를 같은 프로세서 패키지에 속하는 모든 코어들이 공유하고 있다. 따라서 0번 코어와 L3 캐쉬를 공유하는 1, 2, 3번 코어들에게 통신 프로세스의 친화도를 설정한 경우 (즉, <0-1>, <0-2>, <0-3>) 거의 동일한 자원 요구를 보이고 있다. 그리고 NUMA의 경우는 메모리가 각 프로세서에 분산 연결되기 때문에 메모리의 지역성 또한 성능에 영향을 미친다. 그 결과 SMP의 경우와는 다르게 친화도를 서로 다른 프로세서 패키지의 코어에 각각 할당했을 때 (<0-4>, <0-5>, <0-6>, <0-7>의 경우) 자원 요구량이 그렇지 않은 경우에 (<0-1>, <0-2>, <0-3>) 비해서 5% 이상 증가하는 것을 알 수 있다.

3. 결과 활용 방안 및 토의

지금까지 살펴 본 측정 결과는 프로세서 친화도에 따라 통신 처리율은 크게 변화가 없을 수는 있어도 CPU 자원의 요구량에는 크게 영향을 준다는 것을 시사하고 있다. 이러한 현상은 기가비트 이더넷 보다 큰 대역폭을 제공하는 네트워크 연결의 경우는 더욱 심화될 것이다. 따라서 프로세스 스케줄러는 인터럽트의 프로세서 친화도를 고려하여 통신 프로세스의 스케줄링을 결정해야 할 것이다. 이

때, 필요한 중요한 정보는 현재 네트워크 인터럽트의 프로세서 친화도와 멀티코어의 캐쉬 공유 구조이다. 현재의 인터럽트 프로세서 친화도는 /proc 파일 시스템의 smp_affinity 파일 정보를 보유하고 있는 irq_desc 커널 구조체를 통해서 알 수 있다. 그리고 멀티코어의 캐쉬 공유 구조는 /sys 파일 시스템의 shared_cpu_map 파일을 통해서 알 수 있다.

이들 정보를 이용해서 통신 프로세스를 인터럽트 친화도가 설정된 코어에 스케줄링하여 프로세서 자원을 절약하는 효과를 얻을 수 있다. 프로세서 친화도를 통한 통신 성능 향상과 관련된 연구들이 진행되었으나 [10][11], 프로세서 친화도를 인터럽트와 프로세스 부분으로 구분하고 캐쉬 공유 구조가 복잡한 상황에 대해 고려한 연구 결과는 아직 기록되지 않고 있다. 하지만 이러한 연구를 위해서 고려해야 하는 부분들은 무척 많다. 우선 인터럽트의 프로세서 친화도를 설정한 코어에 프로세스를 스케줄링하기 위해서는 프로세스 이동(migration) 오버헤드를 고려해야 한다. 아울러 캐쉬 공유 구조를 고려하여 최적에 가까운 코어에 스케줄링 하는 방안 또한 고려할 수 있다. 어려운 문제 중 하나는 응용 프로세스가 네트워크 I/O뿐만 아니라 디스크 I/O도 수행할 경우는 디스크 I/O 인터럽트의 프로세서 친화도 또한 고려해야 한다. 이와 같은 사항들을 고려하여 프로세스 스케줄링이 수행될 경우 높은 대역폭의 네트워크를 사용하는 서버는 상당히 많은 프로세서 자원을 절약할 수 있을 것으로 기대된다.

4. 결론 및 향후계획

본 논문은 SMP와 NUMA 구조의 멀티 코어 프로세서 환경에서 통신 프로세스와 네트워크 인터럽트의 프로세서 친화도를 변화시키며 네트워크 처리율과 코어의 유휴 자원 양을 분석하였다. 분석 결과 프로세서 친화도에 따라 네트워크 처리율의 변화는 미비하나 코어의 유휴 자원 양은 현저한 차이를 보이는 것을 알 수 있었다. 또한 코어 간의 캐쉬 공유 구조 및 메모리 분산 구조에 따라 그 성능에 차이가 있음을 정량적으로 보였다. 이와 같은 분석 결과는 멀티 코어의 특성을 고려한 프로세스 스케줄러를 구현하기 위해서 필수적인 데이터로서 활용될 수 있다.

향후 연구로서 이러한 멀티 코어의 특성을 고려하여 현재 개발되고 있는 리눅스 프로세스 스케줄링 기법을 완성하고 그 성능을 측정하려고 한다. 현재 개발되고 있는 프로세스 스케줄링 기법은 프로세스의 통신 집중도, 프로세서의 부하 등을 고려하여 최적의 코어에 프로세스를 할당한다. 또한 여러 개의 네트워크 인터페이스를 장착하고 있는 서버 노드에 적합한 프로세스 스케줄링 정책도 고려하고 있다.

참고문헌

[1] IEEE, IEEE Std 802.3ae-2002, Media Access Control (MAC) Parameters, Physical Layers, and Management Parameters for 10Gbps Operation, August 2002.

[2] InfiniBand Trade Association, http://www.infinibandta. org.

[3] Myricom Inc., http://www.myri.com.

[4] H.-W. Jin, Y.-J. Yun and H.-C. Jang, "TCP/IP Performance Near I/O Bus Bandwidth on Multi-Core Systems: 10-Gigabit Ethernet vs. Multi-Port Gigabit Ethernet," In Proc. of International Workshop on Parallel Programming Models and Systems Software for High-End Computing (P2S2), September 2008.

[5] A. Foong J. Fung, and D. Newell, "An In-Depth Analysis of the Impact of Processor Affinity on Network Performance," In Proc. of IEEE International Conference on Networks (ICON 2004), pp. 244-250, November 2004.

[6] T. Scogland, P. Balaji, W. Feng and G. Narayanaswamy, "Asymmetric Interactions in Symmetric Multi-core Systems: Analysis, Enhancements and Evaluation," In Proc. of IEEE/ACM International Conference for High Performance Computing, Networking, Storage and Analysis (SC), November 2008.

[7] 윤대석, 박희권, 최종무, "NUMA 환경에서 메모리 친화력을 고려한 부하 균등 모델", 한국컴퓨터종합학술대회 논문집, 2008.

[8] Intel Multi-Core, http://www.intel.com/multi-core/index.htm.

[9] AMD Multi-Core, http://multicore.amd.com/.

[10] J. D. Salehi, J. F. Kurose, and D. Towsley, "The Effectiveness of Affinity-Based Scheduling in Multiprocessor Network Protocol Processing," IEEE/ACM Transactions on Networking, 4(4):516-530, August 1996.

[11] P. Willmann, S. Rixner, and A. Cox, "An Evaluation of Network Stack Parallelization Strategies in Modern Operating Systems," In Proc. of the Annual Conference on USENIX '06 Annual Technical Conference, 2006.