멀티코어 NUMA 시스템에서 메모리 병렬성이 성능에 미치는 영향

Effects of Memory Parallelism on Performance in Multicore NUMA Systems

저자
(Authors)

윤준기, 최용재, 최종무
Yoon Junkee, Choi Yongjae , Choi Jongmoo

출처
(Source)

한국정보과학회 학술발표논문집 , 2016.6, 64-66 (3 pages)

발행처
(Publisher)

한국정보과학회
KOREA INFORMATION SCIENCE SOCIETY

URL

http://www.dbpia.co.kr/Article/NODE07018110

APA Style

윤준기, 최용재, 최종무 (2016). 멀티코어 NUMA 시스템에서 메모리 병렬성이 성능에 미치는 영향. 한국정보과학회 학술발표논문집, 64-66.

멀티코어 NUMA 시스템에서 메모리 병렬성이 성능에 미치는 영향

윤준기 ○ 최용재 최종무

단국대학교

jun6826@naver.com, bestjaee@gmail.com, bicology@gmail.com

Effects of Memory Parallelism on Performance in Multicore NUMA Systems

Yoon Junkee○ Choi Yongjae Choi Jongmoo

Dankook University

요 약

최근 컴퓨터 시스템에서는 프로세스들의 메모리 접근 병목 현상을 줄이기 위해 NUMA(Non-Uniform-Memory-Access) 구조를 사용하고 있다. NUMA 구조는 지역 메모리와 원격 메모리에 접근 성능이 큰 차이를 보이기 때문에 기존 운영체제에서는 프로세스의 메모리 할당을 지역 메모리 내에서 할당한다. 이로 인해 마이크로 아키텍처 자원의 경쟁이 발생하며, 지역 메모리 내에서만 접근을 하기 때문에 메모리 접근 병렬성이 저하된다.[2] 본 논문에서는 메모리 접근 병렬성과 마이크로 아키텍처 자원 경쟁에 따른 성능 영향력을 분석하기 위해 연구를 진행하였다. 이를 위해 다섯 가지 실험 환경에 대해서 실험 및 관찰을 하였으며, PMU(Performance Monitoring Unit)[6]에 기반한 마이크로 아키텍처 자원 사용량 측정을 통해 메모리 병렬성이 시스템 성능에 미치는 영향을 증명하였다. 그 결과 지역/원격 메모리가 성능에 미치는 영향보다 메모리 병렬성이 성능에 미치는 영향이 훨씬 크게 나타났다.

1. 서 론

최근 컴퓨터 시스템에서의 NUMA 환경은 메모리 병렬성 극대화 기반의 성능 향상을 목표로 지속적으로 발전하고 있다. 이러한 추세를 바탕으로 IBM X3850은 Intel사의 Xeon® 프로세서를 장착하여 총 128코어를 활용하고 있으며, AMD Bulldozers는 Opteron® 프로세서를 장착하여 총 64개의 코어를 구축하고 있다. 이와 같이 하드웨어 수준의 발전과 더불어 운영체제도 자원의 이용을 효율적으로 지원하고자 가상화 기술, 클라우드 컴퓨팅, 빅데이터등 끊임없는 연구가 진행되어 왔다. 그림1은 NUMA 아키텍처 구조로써 각각의 소켓에는 여러 개의 코어들이 있고 하나의 지역 메모리와 연결 되어있다. 각각의 소켓은 QPI(Quick Path Interconnect)로 연결 되어있다. NUMA 환경 기존 연구들은 프로세스의 성능 향상을 위해 지역 메모리로부터 메모리를 최대한 할당 받도록 하였다.[1] 하지만 멀티코어 내부의 마이크로 아키텍처 자원 경쟁으로 인해 다수의 프로세스가 지역 메모리에만 접근하는 성능 병목 현상이 일어났다. 이를 위해 우리는 마이크로 아키텍처 자원 경쟁을 고려한 메모리 병렬성에 따른 성능 향상 모델을 연구하였고, 다섯 가지 실험에 대해서 실험 및 관찰을 하였다. PMU를 통해 마이크로 아키텍처 자원 사용량 측정을 통해 메모리 병렬성이 시스템 성능에 미치는 영향을 증명하였다. 또한 SPEC CPU 2006[5] 벤치마크의 여러 자원을 사용하여 실험의 정확성을 높이고자 하였다. 본 논문에서 2장에서는 마이크로 아키텍처 요소들에 대한 설명을 한다. 3장에서는 NUMA 환경에서 코어와 메모리 노드 배치에 따른 서로 다른 다섯 가지 실험 환경에 대해 설명을 한다. 4장에서는 실험 환경에 따른 SPEC CPU 2006 벤치마크의 성능 비교 및 분석을 한다. 마지막으로 5장에서는 실험에 대한 결론을 짓는다.

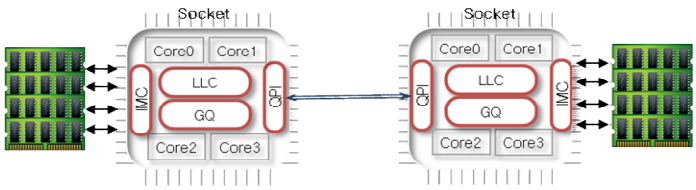

그림1 NUMA 아키텍처 구조

2. 마이크로 아키텍처 자원

NUMA 시스템의 CPU 소켓 안에는 코어와 아키텍처 요소들로 구성 되어있다. 그림2과 같이 LLC, GQ, IMC, 그리고 QPI등의 마이크로 아키텍처 요소들로 구성되어 있다. 이러한 요소들은 하나의 코어에 종속되어있는 L1, L2 캐쉬와 같은 자원과는 다르게 코어들에 의해 공유되는 자원으로 이용된다. LLC(Last Level Cache)는

여러 코어들에 의해 공유되어 정보를 입출력 할 수 있으며, GQ(Global Queue)는 주소 맵핑 방식에 기초하여 LLC, IMC, 그리고 QPI 정보를 요청하는 역할을 한다. IMC(Integrated Memory Controller)는 지역 메모리를 관리하며, 각각의 소켓의 QPI는 다른 소켓의 지역 메모리와 LLC 접근에 사용한다. 이러한 마이크로 아키텍처 자원들은 멀티코어 환경에서 코어들이 동시에 수행하면 할당을 위한 경쟁이 발생하고, 이로 인해 마이크로 아키텍처 자원 갑섭 현상이 발생하여 성능 저하로 이어진다. 예를 들어 NUMA 구조에서 하나의 코어가 LLC에 접근 하는 경우라면, 이로 인해 다른 코어들과 LLC 자원 경쟁을 하게 된다.

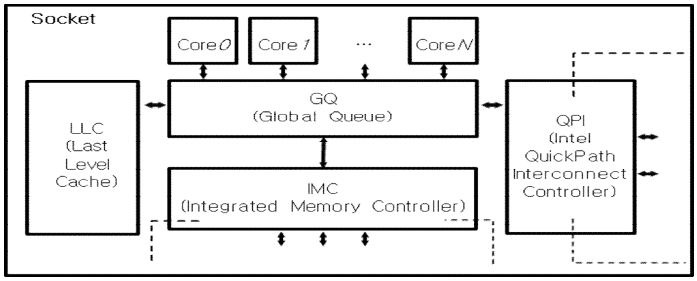

그림2 마이크로 아키텍처

3. 시스템 성능 비교를 위한 다섯 개의 실험 환경

메모리 접근 병렬성과 마이크로 아키텍처 자원 경쟁에 따른 성능 영향력을 분석하기 위해 그림3과 같이 다섯 개의 서로 다른 실험 환경을 구성하였다. 이와 같은 환경 설정은 지역/원격 메모리 접근에 따른 성능, LLC의 개수에 따른 성능, 그리고 메모리 병렬성에 따른 성능을 분석하기 위함이다.[4] 그림3에서 ①은 실험 환경은 한 소켓의 네 개의 프로세스가 모두 수행되며 지역 메모리에만 참조 한다. ②는 ①과 같이 한 소켓의 모든 프로세스가 사용 되지만 지역 메모리와 원격 메모리를 사용하였으며, 리눅스 numactl[3] 유틸리티를 사용하여 50:50으로 메모리를 나누어서 배치하였다. 이 구조는 메모리 병렬성에 대한 성능을 알아보기 위함이다. ③은 두 개의 소켓을 모두 사용하고, 각각의 소켓의 모든 코어들 중 두 개의 코어만 프로세스에 할당하고 메모리는 첫 번째 소켓의 메모리만 사용하는 구조로써, LLC 개수에 따른 성능 측정을 하기 위함이다. ④는 ③과 비슷하지만 프로세스들이 지역 메모리와 원격 메모리에 모두 접근하는 방식이다. 마지막으로 ⑤는 하나의 소켓에서 모든 프로세스들이 수행되며, 다른 소켓의 메모리를 접근, 즉 원격 메모리에 접근하는 방식이다. ⑤는 원격 메모리에 접근함에 따라 발생되는 성능을 분석하기 위한 ①과 대조되는 실험이다. 지역 메모리에 접근하는 비중은 ①이 가장 크고, ⑤번이 가장 적다. LLC경우 ③, ④가 LLC를 두 개를 사용하기 때문에 나머지 세 개의 환경에서 LLC를 한 개를 사용했을 경우와 비교하여 성능에 영향을 끼치는 점을 파악할 수 있다. 메모리 병렬성은 두 개의 메모리를 사용하는 ②, ④가 메모리 노드를 두 개를 사용하기 때문에 다른 실험 환경들에 비해 IMC, QPI와 같은 마이크로 아키텍처 자원에 대한 경쟁이 적게 발생한다. 따라서 ②, ④가 병렬성이 좋게 된다.

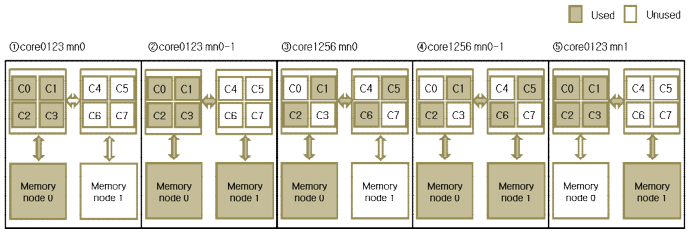

그림3 다섯 개의 실험 환경

4. 다섯 개의 실험 환경에 따른 SPEC CPU 2006 벤치마크 성능 비교 및 분석

실험을 위해 두 개의 Intel Xeon X5570 프로세서를 사용하였고, 각각의 프로세서는 8MB LLC와 두 개의 16GB 지역 메모리, 총 32GB 메모리로 실험을 진행하였다. 운영체제는 리눅스 커널 버전 4.0.3에서 실험을 진행하였다. 우리는 실험에서 네 개의 프로세스를 사용하였고, SPEC CPU 2006 벤치마크의 여덟 개의 어플리케이션을 사용하였다. 그림4는 각 실험 환경에 따른 수행 시간 결과를 보여준다. mcf 어플리케이션을 제외한 모든 어플리케이션들에서 ②와 ④의 성능이 다른 실험 환경보다 월등하게 성능이 좋게 나타났다. 이를 바탕으로 메모리 병렬성이 다른 요소들이 성능에 미치는 영향보다 큰 것을 알 수 있다. ⑤는 ①과 비교를 하면 원격 메모리와 지역 메모리의 성능 차이를 보여준다. 또한 LLC 개수가 두 개인 ④에서 LLC가 한 개인 ②보다 성능이 더 좋게 나오는 걸 볼 수 있다. 우리는 마이크로 아키텍처 자원들과 성능간의 상호관계를 분석하기 위해 PMU(Performance Monitoring Unit)이라는 하드웨어를 사용하여 측정하였다.

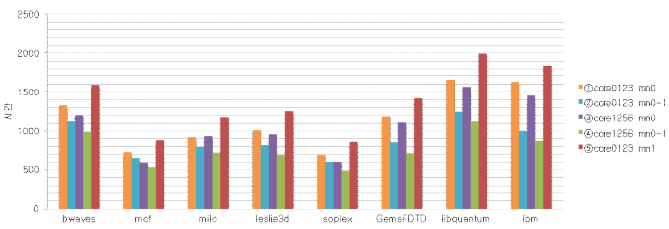

그림4 SPEC CPU 실험 결과

측정한 마이크로 아키텍처 요소들은 표1에 나타나 있다. LLC의 접근 횟수와 LLC의 미스 율을 측정함으로써

LLC의 자원 이용 량을 측정하였고, 메모리 대역폭을 측정하기 위해 IMC의 읽기/쓰기를 수행한 횟수를 측정하였다. 그리고 지역/원격 메모리에 따른 성능을 측정하기 위해 QPI가 지역/원격 메모리에 읽기/쓰기를 요청한 횟수를 측정하였다.

표1. PMU 이벤트

Performance Monitoring Events	Descriptions	Deriving Information
LLC_REFERENCE	Last Level Cache accesses	LLC
LLC_MISSES	Last Level Cache misses	
UNC_IMC_NORMAL_READS.ANY	IMC normal read requests	Memory bandwidth
UNC_IMC_WRITES.FULL.ANY	IMC write request	
UNC_QHL_REQUESTS.REMOTE_READS	QPI Home Logic remote read requests	NUMA (Remote Memory)
UNC_QHL_REQUESTS.REMOTE_WRITES	QPI Home Logic remote write requests	
UNC_QHL_REQUESTS.LOCAL_READS	QPI Home Logic local read requests	NUMA (Local Memory)
UNC_QHL_REQUESTS.LOCAL_WRITES	QPI Home Logic local write requests	

표2는 lbm 어플리케이션을 PMU로 측정한 결과이다. 이러한 결과를 통해 ①, ③ 두 개의 결과와 ②, ④ 두 개의 결과를 서로 비교해보면 LLC 적중 율이 좋아질수록 성능이 좋아지는 척도로 사용할 수 있지만, 실험에서 LLC 참조 횟수가 적었기 때문에 성능에 큰 영향을 미치지 않는다는 것을 알 수 있었다. 지역 메모리에 접근하는 비중은 ①이 가장 크고, ⑤번이 가장 적기 때문에 ①의 성능이 더 좋게 나왔다. 따라서 지역/원격 메모리 접근에 따라 시스템 성능에 큰 영향을 미치는 것으로 보인다. 그러나 시스템 성능에 제일 큰 영향을 미친 것은 메모리 병렬성 이다. ②, ④의 메모리 대역폭이 다른 실험 환경들에 비해 두 배정도 크게 나왔다. 그리고 이러한 차이는 SPEC CPU 실험에서도 ②, ④의 성능이 다른 실험 환경과 비교해서 월등히 좋았다.

표2. lbm어플리케이션 PMU 분석 결과

lbm	LLC Hit Ratio	LLC Reference	Local Request of QHL	Remote Request of QHL	MBW	Execution Time
① configuration	0.149064	9.92E+10	2.69E+11	1.55E+08	2256.451	1649.955
② configuration	0.20388	6.58E+10	1.36E+11	1.34E+11	3757.577	1006.341
③ configuration	0.241596	8.32E+10	1.48E+11	9.33E+10	2261.483	1439.691
④ configuration	0.245581	5.43E+10	1.32E+11	1.31E+11	4052.137	868.2845
⑤ configuration	0.229465	7.02E+10	1.24E+09	2.68E+11	2044.691	1831.024

5. 결 론

최근 컴퓨터 시스템에서 NUMA 환경을 통해 메모리의 병렬성이 높아졌다. 다수의 코어들이 동시에 마이크로 아키텍처 자원을 경쟁하며, 사용하기 때문에 시스템 성능 병목 현상이 생겼다. 그래서 우리는 코어 사용 개수 그리고 메모리 노드 배치로 이러한 문제점을 완화시키고자 서로 다른 다섯 개의 실험 환경을 만들었다. PMU를 사용하여 마이크로 아키텍처 자원들과 성능간의 상호관계를 분석 하였다. 그 결과, LLC 적중율에 따라 성능이 좋아지긴 하지만, LLC 참조 횟수가 적어 성능에 큰 영향을 미치지 않았다. 지역/원격 메모리 접근에 따른 시스템 성능에 중요한 역할을 했지만, 메모리 병렬성에 따를 성능 변화가 너무 커서 실험 결과에 큰 영향을 미치지 못했다. 메모리 노드 두 개를 사용한 실험 환경에서는 한 개를 사용한 실험 환경보다 메모리 대역폭이 두 배 이상 차이가 나는 것으로 나타났다. 본 논문의 실험을 통해 NUMA 환경에서 소켓에서 프로세스가 수행될 때 메모리 병렬성이 제일 큰 영향을 미치는 것을 잘 보여주고 있다.

6. 참고 문헌

[1] S. Boyd-Wickizer, “COREY: AN OPERATING SYSTEM FOR MANY CORES". In 8th USENIX Symp. On Operating Systems Design and Implementation

[2] Hood, Robert, et al. "Performance impact of resource contention in multicore systems." Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on. IEEE, 2010.

[3] numactl(8) - Linux man page
http://linux.die.net/man/8/numactl

[4] T. Brecht, “On the importance of parallel application placement in NUMA multiprocessors". In 4th Symp. Experiences with Distributed & Multiprocessor Syst., pp. 1{18, USENIX, Sep 1993.

[5] SPEC CPU2006 Benchmark Descriptions
https://www.spec.org/cpu2006/publications/CPU2006benchmarks.pdf

[6] Performance Analysis Guide for Intel®Core™ i7 Processor and Intel® Xeon™ 5500 Processors
https://software.intel.com/sites/products/collateral/hpc/vtune/performance_analysis_guide.pdf

이 논문은 2016년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (R0190-15-1085, 1,000Core 이상 Scale Out 가능한 클러스터 데이터베이스 플랫폼 개발)