CC-NUMA 시스템에서의 프로세싱 노드간 네트워크 부하 분석

김 태균 왕 희정 이 강우
동국대학교정보통신공학과
{tgkim, zodiac, klee}@dgu.ac.kr

Analysis of Network Communication Overhead Among Processing Nodes in CC-NUMA System

Taegyoun Kim Heejung Wang Kangwoo Lee
Dept. of Information and Communication, Dongguk University

요 약

CC-NUMA 시스템은 SMP 시스템의 장점인 프로그래밍의 편리함, 작업 환경의 유연함 및 관리의 용이함등을 유지하는 한편, SMP의 단점이었던 확장성까지 제공한다. 더욱이 메모리 장벽 즉 급격히 빨라지는 프로세서의 처리 속도에 비해 메모리의 속도는 거의 변화가 없음으로 인하여 야기되는 문제를 극복할 수 있는 구조적인 대안으로 각광 받고 있다. 이러한 CC-NUMA 시스템은 노드간의 논리적인 거리가 길기 때문에 프로세싱 노드간의 통신이 시스템의 성능에 영향을 미치는 가장 핵심 요소가 된다. 따라서 노드간의 통신을 최소화 해주기 위한 노력으로 각 노드에 장착되어지는 원격 캐쉬의 중요성이 강조된다.

본 논문에서는 CC-NUMA 시스템에서의 노드간 데이터 통신의 유형을 파악하고, 원격 캐쉬의 블록 사이즈에 따른 이들이 발생횟수의 변화를 분석하였다. 인스트럭션 시뮬레이터인 CacheMire와 SPLASH II 벤치마크 중 하나인 FFT를 이용하여 실행-구동 시뮬레이션을 통해 원격캐쉬 블록의 크기가 증가할수록 노드간 통신의 횟수는 물론 전송되는 데이터의 절대적인 양이 감소한다는 사실을 알 수 있었다.

1. 서론

근래들어 마이크로 프로세서들이 SMP 시스템을 구성하는데 용이하도록 제작되고있다. 가장 대표적인 예가 Intel 사의 Pentium 계열 프로세서들로써, 4개의 프로세서를 장착한 SHV 보드를 통해 SMP가 지원되는 하나의 프로세싱 노드로 손쉽게 구현 할 수 있게 되었다. SMP 시스템은 편리한 프로그래밍 환경과 사용 및 관리의 효율성을 비롯한 여러 가지 장점을 가지고 있으나 확장성의 취약함으로 인하여 그의 활용이 제한적이다. 현재는 이러한 SMP 시스템의 장점을 그대로 유지하는 한편 확장성을 제공하는 구조인 CC-NUMA 시스템이 각광을 받고 있다.

현재 상용 벤더들이 채택하고 있는 CC-NUMA 시스템의 일반적인 구조는 SHV 보드를 통해 이루어진 SMP 시스템을 하나의 프로세싱 노드로 정의하고 여러개의 노드들을 SCI등의 고속 네트워크로 연결한 형태를 갖는다. 이러한 형태의 CC-NUMA 시스템은 빠른 속도로 다중 프로세서 시장을 지배할 것으로 기대되어지고, 따라서 일반적인 CC-NUMA 시스템의 구조 및 성능적인 특성을 면밀히 분석하고 평가해야하는 필요성이 제기된다[6].

일반적으로 어떠한 프로그램을 실행하는 다중 프로세서 시스템의 성능은 프로세서들이 계산을 수행하는데 소요되는 *프로세싱시간*과 데이터를 공유함으로써 발생하는 프로세서간의 *통신시간*의 합으로 나타내어 진다. 프로세싱시간은 주어진 프로그램의 고유한 특성으로 간주되나 통신시간은 사용되는 시스템의 구조적인 특성에 의하여 결정되므로 이를 흔히 통신 오버헤드라 일컫는다. 따라서, 다중 프로세서에 관련된 연구의 대부분은 이러한 통신 오버헤드를 최소화하는 것을 목적으로한다.

SMP의 경우 여러 가지 장점을 지녔음에도 불구하고 프로세서의 수가 늘어날수록, 또한 프로세서의 클럭 속도가 빨라질수록 프로세서간의 통신의 빈도가 급속히 증대되어 상호연결망인 버스에 병목현상이 야기되어 통신 오버헤드가 전체적인 성능에 치명적인 영향을 미치게 된다. 이러한 문제는 CC-NUMA 구조를 택함으로써, SMP의 버스에 부과되었던 통신 부하를 여러개의 프로세싱 노드의 내부 버스들로 분산시켜 병목현상을 제거하고 통신 오버헤드를 급감시킬수 있다. 그러나, SMP 시스템에는 없었던 *프로세싱 노드간의 통신*이라는 새로운 현상이 야기된다. 특히, 프로세싱 노드간의 통신은 노드간의 논리적인 거리가 길어 부주의한 설계는 자칫 더 많은 통신 오버헤드를 초래할 수도 있으므로 이에 대한 면밀한 관찰과 연구가 수행되어져 왔다.

이러한 연구의 결과 중 하나인 원격캐쉬(RAC: Remote Access Cache)는 노드간의 데이터 전송을 최소화하기 위한 노력의 일환으로 연구되었다[5]. 본 연구에서는 목표 시스템에서의 데이터 전송 유형을 파악하는 한편, 원격 캐쉬의 역할과 효용에 대해 보다 면밀히 조사하여, 전체적인 시스템의 성능에 미치는 영향을 파악하여 본다.

본 논문의 2장에서는 CC-NUMA 시스템과 관련된 기존의 연구들을 정리하고, 본 연구의 목표 시스템인 확장 MESI를 사용하는 CC-NUMA 시스템과 이러한 시스템에서 발생할 수 있는 데이터 통신의 유형을 3장에서 분석 및 정리한다. CacheMire에 기반한 수행-구동 시뮬레이션 환경 및 시뮬레이션 결과를 4장에서 제시하고, 본 연구의 결론을 마지막 장에서 내린다.

2. 관련 연구

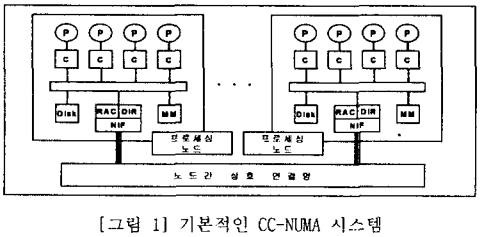
현재 연구 및 개발되어지고 있는 CC-NUMA 시스템은 표준화된 상호접속 기법에 의해 Intel의 SHV보드와 통합한 다중 프로세서 시스템으로써, SMP 시스템과 같이 공유 메모리 응용 프로그램의 실행이 가능하다. 하지만, 원격지의 메모리에 있는 데이터의 접근에는 노드내 메모리 접근에 비해 많은 지연 시간이 요구되어진다. 따라서 이로 인해 발생하는 시스템의 성능저하를 최소화하기 위해 대부분의 CC-NUMA시스템은 원격캐쉬를 각 노드에 위치시키고, 다른 노드에서 읽어 온 데이터를 보관하게 된다.

원격 캐쉬에 관한 대표적인 연구로써 DASH 시스템의 경우 각 노드는 스누핑에 기반한 버스 구조를 사용하였다. 캐쉬 일관성 유지는 full-bit-vector 디렉토리에 기반하였으며, 원격 캐쉬를 각 노드에 두었다[3]. Sequnet에서 발표한 STiNG의 경우 노드간 연결은 SCI링을 사용하였으며, 32Mbyte의 원격 캐쉬를 두고 디렉토리를 이용한 일관성 유지방법을 사용하였다[4]. NUMAchine 시스템에서는 다이렉트 맵핑 방식으로 일관성 유지가 이루어지는 4MByte의 원격 캐쉬를 두었다[2]. Encore의 GigaMax는 2단계의 계층 버스를 사용한 시스템으로 전역 버스와 클러스터 내의 지역 버스에 모두 스누핑에 기반한 방법을 이용하였다. 현재 출시된 상용제품으로는 IBM사의 NUMA-Q[9], Data General사의 AViiON servers[10], SGI사의 Orgin Server[11]등을 들 수 있다.

3. 목표시스템

3.1. 기본적인 CC-NUMA 시스템

현재 개발되어지고 있는 많은 CC-NUMA 시스템들은 [그림 1]과 같은 구조로서 Intel사의 SHV 보드를 기반으로 하고 있다. SHV 보드는 Intel Xeon 프로세서가 4개 장착된 시스템으로, 프로세서들간의 캐쉬 일관성은 MESI 정책에 의해 관리되어진다. CC-NUMA 시스템에서는 SCI등의 고속의 네트워크가 이들 프로세싱 노드를 연결하여, 확장성을 제공한다.



[그림 1] 기본적인 CC-NUMA 시스템

본 연구에서는 CC-NUMA 시스템의 캐쉬 일관성 정책으로 SHV에서 적용되는 MESI를 적용하며, 또한 원격캐쉬에 대해서도 이를 확장 적용하는 확장 MESI를 구현한다.

3.2. 데이터 통신의 유형

목표 시스템에서 가장 기본적인 프로세서간 통신의 발생 요인이 되는 캐쉬 접근 실패는 지역 캐쉬 접근 실패와 원격 캐쉬 접근 실패로 구분된다. 또한, 쓰기 작용에의한 무효화(Invalidation) 신호 역시 통신의 원인이 된다. 이러한 통신을 유발하는 원인 중에서 가장 많은 양의 데이터 통신을 야기하는 경우는 각 경우마다 되쓰기를 동반한 경우이다. 이는 지역 캐쉬에 비해 2~4배 정도 큰 원격캐쉬 캐쉬의 블록을 홈 노드의 주 메모리에 되쓰기 한 후 다시 주 메모리로부터 접근 실패가 발생한 프로세서에게까지 전송되어져야 하는 등 여러 단계의 통신 경로를 거쳐야 하기 때문이다. 이에 대해 아래의 절에서 보다 자세히 알아보기로 한다.

3.3. 되쓰기의 종류

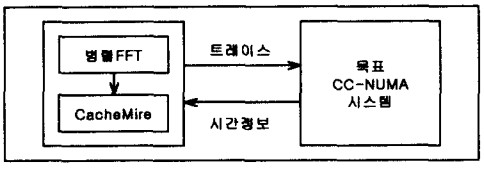
상황을 쉽게 설명하기 위하여 다음과 같은 표현을 쓰기로 한다. *i*, *j*, *k*는 서로 다른 프로세서를 나타내고, *I*, *J*, *K*는 프로세서 *i*, *j*, *k*가 속한 서로 다른 프로세싱 노드를 나타낸다.

- **경우-1**
  프로세서 *i*가 같은 노드의 다른 프로세서의 캐쉬에 수정된(Modified) 상태로 있는 데이터에 대해 접근 실패를 발생한 경우이다. 노드 내부에서 버스를 통한 되쓰기가 필요하다.
- **경우-2**
  프로세서 *i*가 노드 *I*에 할당된 데이터에 대해 접근 실패를 발생하였고, 이 데이터가 노드 *J*의 원격 캐쉬에 수정된 상태로 있는 경우이다.노드 *J*의 원격 캐쉬로부터 노드 *I*의 주 메모리까지 노드간 상호 연결망을 통한 되쓰기가 필요하며, 노드 *I*의 내부 버스를 통한 주 메모리에서 프로세서 *i*의 캐쉬까지의 데이터 전달이 필요하다.
- **경우-3**
  프로세서 *i*가 노드 *J*에 할당된 데이터에 대해 접근 실패를 발생하였고, 이 데이터가 노드 *J*에 속한 프로세서 *j*의 캐쉬에 수정된 상태로 있는 경우이다. 프로세서 *j*의 캐쉬로부터 노드 *J*의 주 메모리까지 노드내부의 버스를 통한 되쓰기가 필요하며, 노드 *J*의 주 메모리로부터 노드 *I*의 원격 캐쉬까지 노드간 상호 연결망을 통한 데이터 전달이 필요하다.
- **경우-4**
  프로세서 *i*가 노드 *J*에 할당된 데이터에 대해 접근 실패를 발생하였고, 이 데이터가 노드 *K*의 원격 캐쉬에 수정된 상태로 있는 경우이다. 노드 *K*의 원격 캐쉬로부터 노드 *J*의 주 메모리까지 노드간 상호 연결망을 통한 되쓰기가 필요하며, 노드 *J*의 주 메모리로부터 노드 *I*의 원격 캐쉬까지 노드간 상호 연결망을 통한 데이터 전달이 필요하다.

4. 시뮬레이션

4.1. 시뮬레이션 환경 및 작업부하

[그림2]는 CacheMire에 기반한 시뮬레이션 환경이다.

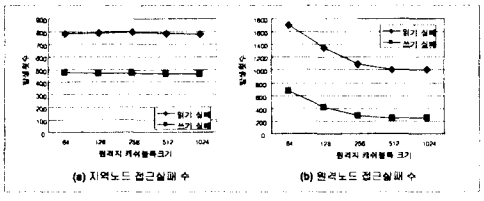


[그림 2] 시뮬레이션 환경

수행-구동 시뮬레이터인 CacheMire은 SPARC 명령어 셋을 수행하여, 읽기/쓰기 동작 및 메모리 주소정보를 포함하는 트레이스를 목표 시스템에게 전달한다[1]. 목표 시스템은 전달받은 트레이스를 이용하여 정의되어진 캐쉬 일관성 정책을 수행하고 그 결과를 보여주게 된다. 시뮬레이션을 위한 작업 부하로는 SPLASH-II의 FFT를 사용하였으며[8], 원격 캐쉬의 블록 크기를 변화시킴으로서 나타나는 캐쉬 실패의 발생 횟수 및 앞서 정의한 4가지 경우의 되쓰기의 빈도를 측정하였다.

4.2. 시뮬레이션 결과

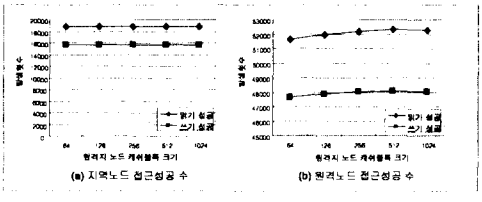
■ 원격 캐쉬 블록 크기에 따른 캐쉬 접근 실패 수



[그림 3] 캐쉬 접근 실패의 수

[그림 3]은 원격 캐쉬의 블록 크기에 따른 캐쉬 접근 실패의 수를 나타낸다. 원격 캐쉬 블록의 크기는 지역 노드내의 데이터 접근 실패에 대해서는 크게 영향을 미치지 않으나, 원격 노드의 데이터에 대해서는 접근 실패의 수가 크게 줄어드는 것을 볼 수 있다.

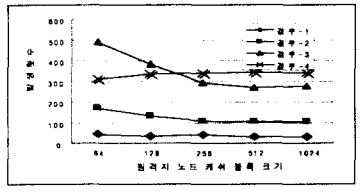
■ 원격 캐쉬 블록 크기에 따른 캐쉬 접근 성공 수



[그림 4] 캐쉬 접근 성공의 수

쓰기 성공은 무효화 신호를 발생하기도 하지만, 수정된 데이터를 생성하는 동작이기도하다. 실험 결과는 원격 캐쉬 블록의 크기가 증가할수록 접근 성공의 수가 조금씩 증가하는 추세를 나타내고 있다.

■ 원격 캐쉬 블록 크기에 따른 되쓰기 수



[그림 5] 4가지 경우의 되쓰기 수

[그림 5]는 원격캐쉬 블록 크기와 관련하여 앞서 정의한 4가지 되쓰기에 대한 결과를 보여주고 있다. 특히 상호연결망을 통해 데이터 전송이 많은 경우-4의 경우, 원격 캐쉬의 블록 크기가 증가함에 따라 되쓰기 수가 감소함을 볼 수 있다. 이는 원격 캐쉬 블록의 증가가 통신양을 감소시키는 요소가 됨을 알려준다.

5. 목표 시스템 성능평가 및 결론

본 연구에서는 Intel SHV 보드를 하나의 프로세싱 노드로한 CC-NUMA 시스템에서의 상호 연결망에서의 통신 발생 요소를 정의하고, 이를 근거로 수행-구동 시뮬레이션을 수행하여, 원격 캐쉬 블록의 크기와 통신 요인과의 관계에 대하여 살펴보았다. 이때 데이터 전송량이 가장 큰 되쓰기를 4가지 유형으로 구별하고, 각 상황에 대해 시뮬레이션을 수행함으로서 실험 결과를 얻었다.

연구의 결과로서 원격 캐쉬 블록 크기의 증가는 캐쉬 접근 실패를 줄이는 역할 뿐만 아니라, 이로 인해 가장 많은 데이터 전송을 필요로하는 경우-4의 되쓰기 수를 역시 감소시키는 효과가 있음을 알 수 있었다.

6. 참고 문헌

[1] M. Brorsson, et al., "The CacheMire Test Bench" , 26th ASS, pp41-49, 1993

[2] S. Brown et al., "The NUMAchine multiprocessor", CSRI-TR-324, CSRI, University of Toronto, 1995.

[3] D. Lenoski, et al., "The Stanford DASH multiprocessor", IEEE Computer,pp 63-79, March 1992.

[4] Tom Lovett, et al.,"STiNG:A CC-NUMA Computer System for the Commercial Marketplace", 23th ISCA pp.308-317, 1996.

[5] E. Moreno, et al., "Efficiency of Remote Access Caches in Future SMP-Based CC-NUMA Multiprocessors: Initial Results" 1997 ISPAN

[6] P. Stenstrom, et al., "Comparative perforamnce evaluation of cache-cohrent NUMA and COMA architectures" 19th ISCA. pp 80-91, 1992

[8] S. Woo, et al., "The SPLASH-2 Programs: Characterization and Methodological Considerations." 22nd ISCA pp24-36, June, 1995.

[9] NUMA-Q Servers, http://www.sequent.com

[10] AViiON Servers, http://www.dg.com

[11] SGI Orgin Servers, http://www.sgi.com