원격 메모리를 이용한 메모리 가상화 서비스 기술

차규일, 김영호, 안신영, 임은지
한국전자통신연구원

요 약

최근 빅데이터 처리에 대한 요구가 급증하면서 매니코어 계산 장치의 개발이 활발히 진행되고 있어 계산 장치와 입출력 저장 장치의 성능 격차는 과거보다 더욱 두드러지고 있다. 이런 상황에서 메모리 가상화 서비스 기술은 입출력 저장 장치의 성능 문제를 완화할 최적의 대안으로 주목받고 있다.

본고에서는 방대한 데이터를 처리해야 하는 응용 프로그램에게 입출력 저장 장치의 성능 한계를 극복하고 데이터 처리 비용을 최소화 할 수 있도록 원격 메모리를 이용한 대용량 가상 물리 메모리 제공 서비스를 지원하는 최근 메모리 가상화 서비스 기술 동향에 대해 알아본다.

Ⅰ. 서 론

최근까지 대규모 데이터를 처리해야 하는 컴퓨팅 분야에서는 파일시스템과 버퍼링(일부 시스템에서는 이를 '공유 캐시'라 부름) 기술을 이용하여 공간과 성능 문제를 해결하고 효과적인 데이터 처리를 수행할 수 있었다. 그러나 매니코어 계산 장치가 일반화된 최근 고성능 컴퓨팅 환경에서 기존에 사용하던 파일시스템과 버퍼링 기술만으로 입출력 성능과 계산 성능의 격차를 해소하고 효과적인 계산 처리 성능을 얻는 것은 어려움이 있다. 뿐만 아니라, 데이터 입출력이 빈번한 응용 프로그램의 경우엔 낮은 입출력 성능이 시스템의 처리 병목을 초래해서 고성능의 계산 장치의 이점을 활용하지 못하고 많은 CPU 처리 시간을 다수 응용 프로그램의 문맥 교환에 소모하는 경향을 보이는 경우도 있다.

초다시점이나 UHD(Ultra High Definition)급 미디어와 같은 실시간 미디어 처리 서비스는 높은 처리 실시간성과 낮은 데이터 재사용성의 특성을 갖는다. 이런 상황은 앞서 설명한 기존 파일시스템과 버퍼링 기술의 한계를 더욱 두드러지게 한다.

최근까지 대규모 데이터를 처리하는 응용 프로그램의 데이터 처리 모델은 '입출력 저장소 기반 선 저장 후처리(Storage-based processing)' 방식을 채택했다. 이 방식은 처리 데이터의 규모가 작거나 처리 실시간성이 낮거나 데이터의 재사용성이 높은 경우는 비교적 높은 성능을 제공할 수 있었다. 그러나 초다시점이나 UHD급 미디어에 기반 한 영상 협업과 같은 경우 더 이상 효과적인 데이터 처리 수단이 될 수 없다.

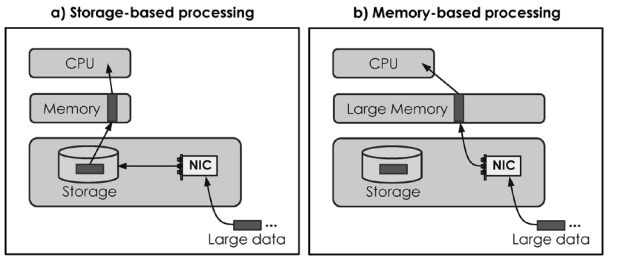

그림 1. 대규모 데이터 처리 모델

초다시점이나 UHD급 미디어 처리의 경우에 고려될 수 있는 데이터 처리 모델은 '대규모 메모리 기반 처리(Memory-based processing)' 방식이다. 이 방식은 데이터 처리에 있어 입출력 저장소를 사용하지 않아 입출력 저장 성능에 영향을 받지 않고 대규모 메모리를 통해 저장 없이 직접 처리하므로 메모리의 성능적인 이점을 얻을 수 있다는 장점이 있다. 그러나 초다시점이나 UHD급 미디어 처리에 대규모 메모리 기반 처리 모델을 채택하기에는 선 저장 후처리를 위한 임시 저장소로 사용하던 입출력 저장소를 대신한 만큼 충분히 큰 대용량 메모리를 갖춘 시스템 구축에 가격적인 경쟁력이 낮아 대규모 데이터 처리 모델의 실적용에 의문이 제기되었다.

초다시점이나 UHD급 미디어는 데이터의 규모(Volume)에 있어 방대할 뿐만 아니라, 생성 속도(Velocity)나 생성 데이터 유형의 다양성 (Variety)에 있어서도 최근 화두가 되고 있는 빅데이터와 맥을 같이 한다. 이런 이유로 단일 서버 중심적인 처리 방식은 현존하는 최고 성능의 계산 장치를 활용하더라도 처리 성능의 물리적인 한계를 극복하기 어려워 다수 노드로 구성된 '분산 컴퓨팅 처리 환경'이 대규모 데이터 처리의 기반 컴퓨팅 인프라로 부각되고 있다.

초다시점이나 UHD급 실시간 미디어를 처리하기 위해 앞서 언급한 분산 컴퓨팅 처리 환경에서 대규모 메모리 기반 처리 모델을 제공하는 분야가 '원격 메모리 가상화 서비스' 기술이다. 여기서 원격 메모리 가상화 서비스란 분산된 노드의 원격 메모리 자원을 통합해 입출력 자원의 물리적 기능 한계를 극복하거나 메모리 자원의 이용률을 극대화하려는 서버 통합 추상화 플랫폼 서비스를 의미한다. 본고에서는 원격 메모리 가상화 서비스 기술에 대해 소프트웨어 구현 방식[1~5]과 하드웨어 구현 방식[6][7] 몇 가지에 대해 집중적으로 살펴보고자 한다.

Ⅱ. 원격 메모리 가상화 구현 수준

클러스터 시스템의 개별 노드 메모리를 통합 가상화해서 메모리 가상화 서비스를 제공하는 원격 메모리 가상화 계층은 구현 수준에 따라 a) 사용자 동작 수준, b) 커널 수준, c) 하이퍼바이저 수준, d) 하드웨어 수준의 네 가지 구현 유형으로 구분된다. 〈그림 2 참조〉 각 구현 방법의 일반적인 장, 단점을 살펴보면 다음과 같다.

첫째, 기존 메모리와 동일한 프로그래밍 시맨틱 제공 여부의 측면에서 볼 때, 사용자 동작 수준 구현법은 실제 물리 메모리를 세밀하게 제어할 수 없는 사용자 영역에서 구현되므로 기존과 동일한 프로그래밍 시맨틱을 제공하기 어렵다. 반면에 커널 수준이나 하이퍼바이저 수준 구현법은 특권 명령을 사용할 수 있는 동작 수준에서 구현됨으로 기존의 프로그래밍 시맨틱을 유지할 수 있다. 또한 하드웨어 수준의 구현법도 시스템 운영체제에 게 실제 메모리 배치의 투명성을 제공하고 데이터 일관성 유지의 책임을 부과하지 않으므로 응용 프로그램에게 기존과 동일한 프로그래밍 시맨틱을 제공하는 것이 가능하다.

둘째, 구현 복잡도 측면에서 볼 때, 사용자 동작 수준 구현법이 디버깅이 용이성으로 인해 구현 복잡도가 낮은 반면 커널 수준이나 하이퍼바이저 수준 구현법이 구현 복잡도가 높다. 하드웨어 수준 구현법은 대상이 되는 하드웨어 규모에 따라 구현 복잡도가 증가하는 경향을 갖는다.

셋째, 시스템의 확장성 측면에서 볼 때, 사용자 동작 수준, 커널 수준 그리고 하이퍼바이저 수준 구현법이 소프트웨어적으로 원격 메모리 가상화 시스템을 구현하므로 대상 시스템 규모를 다계층으로 분할한 후 유지하는 것이 가능해 시스템 확장성이 높은 반면, 하드웨어 수준 구현법은 시스템 규모에 따라 시스템 복잡도가 증가하므로 시스템 확장성을 위한 충분한 설계가 사전에 고려되어야 한다.

넷째, 시스템의 유지보수 측면에서 볼 때, 하드웨어 수준 구

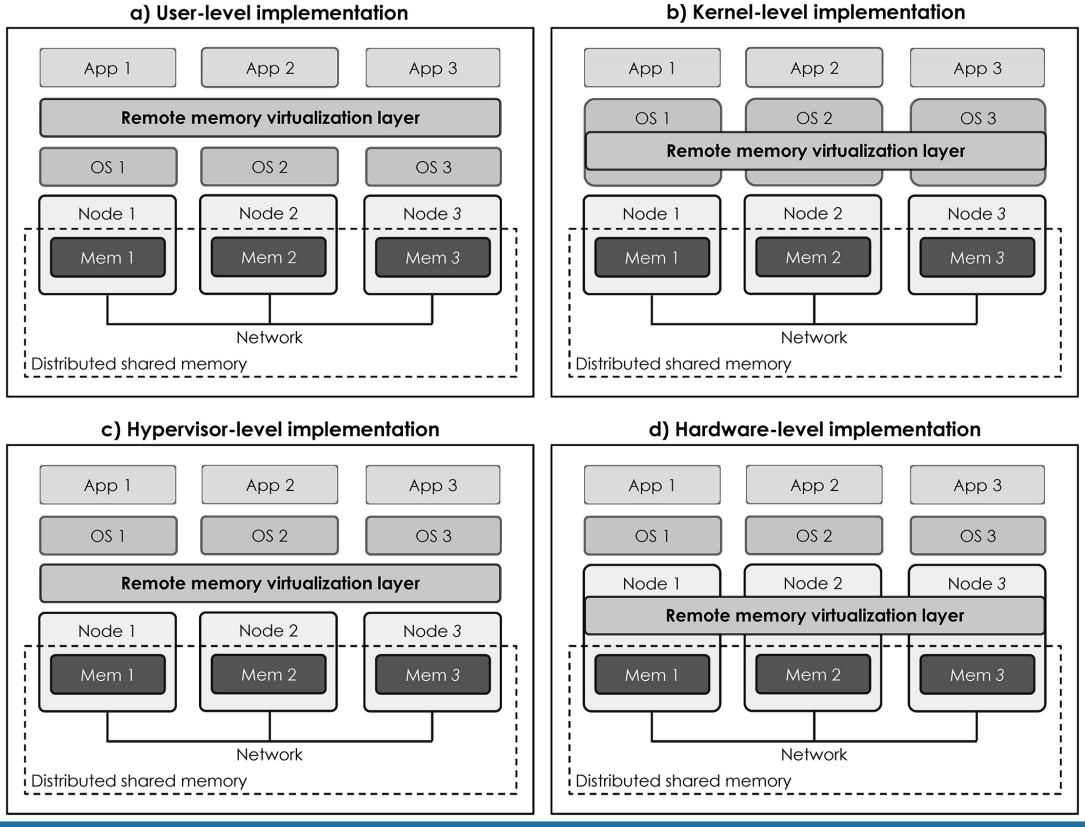

그림 2. 원격 메모리 가상화 계층 구현 유형

현법이 유지보수 비용이 가장 큰 반면, 사용자 동작 수준 구현법의 유지보수 비용이 가장 낮다.

위와 같은 여러 장, 단점으로 인해, 기존에 연구되거나 개발된 많은 원격 메모리 가상화 구현 시스템들은 〈그림 2〉처럼 뚜렷하게 구별되지 않고 서로 혼합된 구현 유형을 보이기도 한다[13][14].

Ⅲ. 소프트웨어 기반 원격 메모리 가상화

소프트웨어 기반의 원격 메모리 가상화 기술은 다양한 방식으로 구현되었다. 이 중에서 순수하게 사용자 동작 수준에서 구현된 Rice 대학의 TreadMarks[1]와 커널 수준에서 구현된 UCLA 대학의 Mirage[2]에 대해 먼저 살펴본 후, 하이퍼바이저 수준에서 구현된 Virtual Iron 사의 VFe[3], New South Wales 대학의 vNUMA[4], ScaleMP 사의 vSMP Foundation[5]에 대해 살펴본다.

여기서 VFe, vNUMA와 vSMP Foundation 기술은 공통적으로 다수의 노드를 소프트웨어 계층으로 추상화한 후 단일 시스템 이미지(SSI: Single System Image)가 운영될 수 있는 SMM(Shared-Memory Multi-processor) 시스템으로 가상화하는 것을 목적으로 한다. 특히, 최근 빅데이터 기술을 처리하기 위해 고성능을 제공하는 단일 서버 개발이 다양한 물리적 요인으로 제한받는 상황에서 Scale-up 기법을 적용해서 다수의 소규모 서버를 통합해 가상화한 후 고성능 시스템을 구축하는 것을 기술의 궁극적 목표로 삼고 있다.

1. TreadMarks

Rice 대학에서 만든 TreadMarks[1]는 SunOS나 Ultrix같은 표준 유닉스 시스템을 위한 분산 공유 메모리 시스템이다. TreadMarks 시스템 이전에도 많은 분산 공유 메모리 시스템들[15]이 존재했으나 실험실 수준을 벗어나지 못했다. 이들이 안고 있던 주된 문제들은 하드웨어 SMM에 의해 사용된 큰 일관성 유지 단위를 사용하던 데이터 일관성 프로토콜을 모방한데서 기인한다. TreadMarks는 기존 DSM(Distributed Shared Memory)의 성능 문제 대부분을 극복한 효율적인 사용자 동작 수준의 원격 메모리 가상화 기술이다.

TreadMarks 구현은 공유 데이터의 접근과 수정을 검출하기 위해 사용자 수준의 메모리 관리 기술만을 사용한다. 초기 DSM의 성능 문제를 극복하기 위해, TreadMarks는 분산 메모리에 존재하는 데이터의 일관성을 유지하기 위해 필요한 통신량의 감소에 주력했다. 기술적으로 false sharing[13]의 영향을 감소시키기 위해 release consistency[17]의 lazy 구현 기법[16]과 다중 쓰기 프로토콜을 사용했다. 또한, 소프트웨어 통신 부하가 높은 응용 프로그램의 성능 향상을 지원하기 위해, TCP/IP 프로토콜 스택을 사용하지 않는 저수준 네트워크 프로토콜 직접 사용하도록 함으로써 네트워크 자체의 부하를 최소화 했으며, 결국 통신 부하가 네트워크 자체보다는 메모리 관리나 데이터 일관성 유지 부하에 좌우되도록 시스템을 구현할 수 있도록 만들었다.

2. Mirage

Mirage[2]는 UCLA 대학에서 UNIX System V IPC 인터페이스와 호환되도록 커널 수준으로 구현된 원격 메모리 가상화 기술이다.

Mirage에서 다수 프로세스에 의해 공유된 영역은 세그먼트로 표현된다. 이 세그먼트는 공유를 필요로 하는 프로세스의 주소 공간에 읽기만 가능(read-only)이나 읽기-쓰기 보호(read-write protection) 속성을 갖고 연결되며 512Kbyte 크기의 페이지들로 분할된다.

Mirage는 공유된 페이지를 세그먼트로 관리하기 위해 Library 사이트, Clock 사이트, 그리고 Requesting 사이트라는 세 가지 site를 운영한다. Library 사이트는 세그먼트를 생성한 사이트로 각 세그먼트의 통제 사이트가 된다. Mirage에서 모든 공유된 페이지에 대한 요청은 이 Library 사이트로 전송되며 순차적으로 처리된다. Library 사이트에는 페이지가 저장된 사이트에 대한 기록을 유지한다. 그러므로 모든 페이

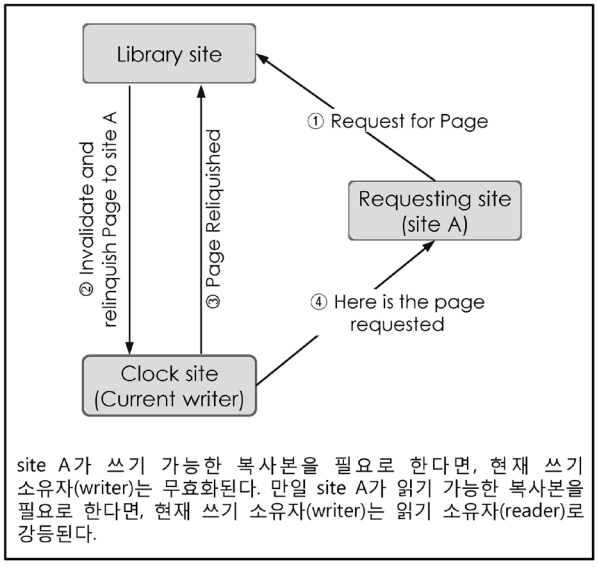

그림 3. Mirage에서 원격 페이지의 폴트 처리

지는 이 사이트를 통해 체크아웃 되어야 만 한다. Mirage에서 Requesting 사이트는 읽기 소유자(reader)이거나 쓰기 소유자(writer)로 기능한다. 쓰기 소유자는 동시에 단 하나의 쓰기 가능한 복사본만을 유지하며, 읽기 소유자는 동시에 다수가 존재 가능하지만 어떤 사이트도 특정 페이지에 대해 읽기 소유자와 쓰기 소유자로 동시에 동작할 수 없다. Clock 사이트는 공유된 페이지의 최신 복사본이 존재하는 사이트를 의미하며 한 페이지의 쓰기 소유자사이트나 다수의 읽기 소유자 중 한 사이트가 해당 페이지의 Clock 사이트로 선택된다.

이상의 세 가지 사이트를 근간으로 공유 세그먼트를 연결한 프로세스의 페이지 테이블(page table) 상의 PTEs(Page Table Entries)는 무효화되고 수정된 커널 폴트 처리기에 의해 〈그림 3〉과 같이 처리되도록 구현되었다. 공유 페이지를 사용하기를 원하는 Requesting 사이트는 Library 사이트에게 페이지 사용권을 요청하고 Library 사이트는 Clock 사이트의 해당 페이지 소유권을 무효화 하고 최종적으로 Requesting 사이트로 요청된 페이지를 반환한다. 이 절차는 기존의 커널의 페이지 폴트 처리기를 수정함으로써 달성할 수 있었고 Mirage는 원격 공유 메모리에 대한 기존 Unix System V IPC 인터페이스와 상위 호환성을 유지할 수 있었다.

3. VFe

Virtual Iron사의 VFe[3]는 분산된 노드의 자원을 가상화해 공유 메모리 다중 프로세서 머신으로 가상화하는 CVMM(Cluster Virtual Machine Monitor) 소프트웨어 계층을 제공한다.

CVMM은 밀결합된 서버 클러스터의 프로그래밍과 사용자의 운영 편의성을 제공하기 위해 소프트웨어적으로 추상화된 가상 서버(Virtual server)로 제공한다. 이러한 가상 서버 각각은 하부 하드웨어 플랫폼의 ISA(Instruction Set Architecture)의 서브셋을 제공(사용자 명령)하거나 일부 특권 ISA에 대해서는 시스템콜 서비스를 제공하는 ViMA (Virtual Iron Machine Architecture)라는 추상 하드웨어 아키텍쳐로 표현된다. 가상 서버는 ViMA 아키텍쳐의 ViMA API를 통해 상위의 Guest OS와 상호 작용하며, 이것은 ViMA API로 Guest OS의 소스 코드를 대체함으로써 가능해졌다.

ViMA는 기존 단일 서버의 하드웨어 구조를 확장해서 하부 레벨의 자원(CPU, Memory, I/O devices 등) 접근을 추상화한다. 이 소프트웨어 계층은 혼합형 가상 머신 모니터[8]로써 구체화되었다. 가상 서버는 하부 하드웨어 아키텍쳐의 ISA를 모두 지원하지 않으므로 내장된 기존 하드웨어 아키텍쳐의 인스턴스가 아니라 다음과 같은 중요 특징을 갖는 ViMA 아키텍쳐의 인스턴스이다.

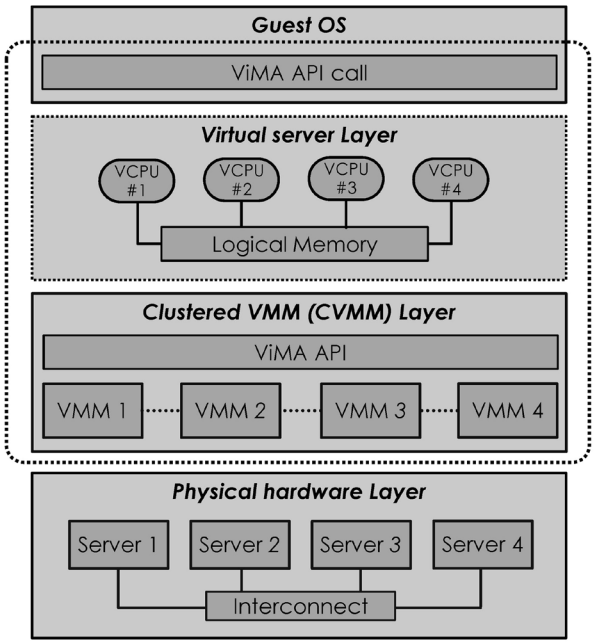

그림 4. Virtual Iron VFe의 추상적 구조

- 가상 하드웨어는 공유 메모리를 갖는 다중 프로세서처럼 동작한다.
- 프로그램은 필요에 따라 가상의 다중 프로세서를 구성하는 모든 물리 서버의 자원을 투명하고 자연스럽게 사용하여 운영된다.
- 각 가상 서버들은 다른 가상 서버들로부터 격리되고 보호된다. 심지어는 가상 서버가 하부의 하드웨어를 공유하는 경우에도 약간의 성능 저하는 발생할 수 있으나 이 특징 자체는 항상 유지된다.

Virtual Iron 사의 VFe는 위에서 언급한 특징을 만족시키기 위해 다중 동작 모드(privilege levels 또는 rings), 응용 프로그램이 특권 수준(privileged level)으로 진입하는 방법, 메모리 재배치와 보호 메커니즘, 그리고 비동기 인터럽트 처리 기능 등의 특징을 제공하는 하드웨어 플랫폼 상에서 동작한다[9]. Intel의 IA32나 x86_64 아키텍쳐는 이런 특징을 제공하는 상용 프로세서로서 VFe는 이런 프로세서를 포함한 소규모 서버를 통합한 클러스터 시스템 환경에서 동작한다.

이런 환경에서 VFe는 ViMA가 제공하는 ViMA API를 사용해 수정된 Linux 커널을 Guest OS로 사용하도록 설계 되어 있으며, 응용 프로그램의 메모리 확장 서비스 사용도 기존 Linux 커널의 메모리 관리 프로그래밍 시맨틱을 동일하게 사용할 수 있도록 구현되었다.

4. vNUMA

New South Wales 대학의 vNUMA[4]는 부하를 최소화하기 위해 분산 통합 하이퍼바이저(VMM: Virtual Machine Monitor)에 DSM 기능을 이식한 유형 1 하이퍼바이저로 Intel Itanium[18] 워크스테이션의 클러스터에 구현되었다. 이 방식은 Disco[19]가 NUMA(Non-Uniform Memory Access) 시스템 상에서 다수의 가상 SMP(Shared Memory Processor) 시스템을 모의하는 것과는 본질적으로 반대 개념이다.

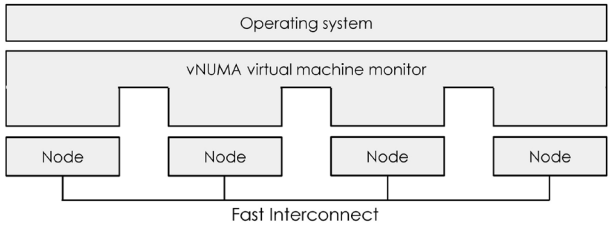

그림 5. vNUMA에서 하이퍼바이저 클러스터링

vNUMA의 VMM은 전가상화(Pre-virtualization) 기법을 사용해서 Linux 커널소스의 변경 없이 커널 바이너리를 생성하기 이전 단계에서 어셈블리 코드의 특권 명령어나 센서티브 명령을 치환하다.

vNUMA에서 제공하는 메모리 관리는 두 가지의 메모리 주소 변환 수준을 거쳐 동작한다. 첫째는 Guest OS가 수행하는 응용 프로그램의 가상 주소(Virtual address)를 가상 머신의 가상 물리 주소(Meta-physical address)로 사상하는 것이고, 둘째는 가상머신의 가상 물리 주소(Meta-physical address)를 호스트 서버의 실제 물리 주소(Physical address)로 사상하는 것이다. vNUMA에서 하이퍼바이저는 가상 물리 주소와 실제 물리 주소의 사상을 관리하며 기존 DSM에서 제공하는 것과 유사한 기능을 제공한다.

vNUMA에서 가상 머신 자체는 모의된 물리 주소 공간(vNUMA에서는 머신 주소 공간 이라고 함)을 갖는다. 이 공간이 vNUMA에서 DSM이 동작하는 레벨이다. 각 머신 페이지는 DSM 시스템에 의해 관리되는 보호 비트와 메타데이터를 갖는다.Guest OS가 가상 주소 사상을 마쳤을 때, DSM 시스템은 요청 보호 비트와 DSM 보호 비트를 바탕으로 해당 가상 주소에 대한 실효 보호 비트를 계산한다. vNUMA는 각각의 머신 페이지에 대해 가상 주소 사상을 추적한다.

vNUMA의 DSM 알고리즘의 구현은 IVY[20]에서 사용되었던 순차적으로 일관적이고 동시 읽기(Multiple reader)와 단독 쓰기(single writer)가 가능한 알고리즘을 사용한다. 일관성 프로토콜을 사용한다. 가상 머신의 머신 페이지들은 클러스터의 각 노드가 페이지들의 일부만을 관리하도록 노드로 나눠 배치된다. 한 노드가 특정 페이지 상에서 폴트될 때, 관리 노드는 첫 번째 인스턴스에 접촉한다. 만일 관리 노드가 소유권을 갖지 않았다면 관리 노드는 소유자에게 이 폴트를 전송한다. 그러면 소유자는 요청 노드로 해당 페이지의 데이터를 직접 copyset과 함께 돌려준다. 데이터를 받은 노드는 필요하다면 임의의 무효화 절차를 수행한다. vNUMA에서는 수정되지 않은 페이지의 데이터에 대한 재전송을 피하기 위해 버전 번호를 사용한다.

vNUMA는 이와 같은 기법을 통해 응용 프로그램에게 분산된 원격 메모리의 기존 메모리와 동일한 프로그램 시맨틱이 제공되는 것을 보장한다.

5. vSMP Foundation

ScaleMP사의 vSMP Foundation[5]은 산업 표준인 다수의 x86 서버를 통합해서 단일한 가상의 고성능 시스템을 제공한다. 주요 목표는 전통적이고 고비용의 SMP나 NUMA시스템을 대체하기 위한 용도로 개발되었으며 소프트웨어 기반의 통합 자원 가상화 플랫폼 vSMP(Versatile SMP) 아키텍쳐 위에 단일 시스템 이미지를 갖는 클러스터 인프라를 제공한다.

vSMP 아키텍쳐는 기성 규격품(off-the-shelf components)을 이용해서 전통적인 다중 프로세서 시스템을 제공한다. vSMP Foundation의 주요 가치는 다중 프로세서 시스템을 생성하기 위해 필수적인 칩셋(chipset) 서비스를 소프트웨어

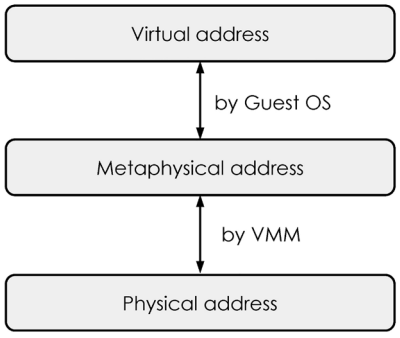

그림 6. vNUMA의 메모리 사상 과정

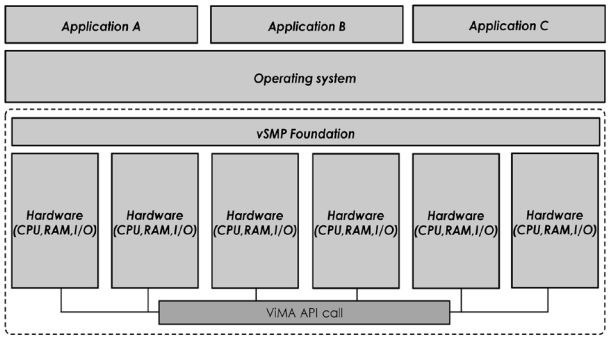

그림 7. ScaleMP의 vSMP Foundation 구조

적으로 제공한다는 것이다. 또한, 운영체제에 의해 요구되는 캐시 일관성 유지 기능, 공유 입출력 기능과 BIOS(Basic I/O System), ACPI(Advanced Con- figuration and Power Interface)와 같은 시스템 인터페이스 등을 제공한다.

현재 vSMP Foundation이 운영되기 위해서는 다음과 같은 요구 사항을 만족해야만 한다.

- 산업 표준 x86 시스템이나 프로세서와 메모리를 갖춘 시스템 보드
- 2노드 이상으로 구성하는 경우는 InfiniBand 네트워크
- 시스템 보드를 vSMP Foundation로 부팅하기 위한 영구 저장 장치

vSMP Foundation이 각 노드 또는 보드로 로드되고 나면, vSMP Foundation은 각 노드의 계산, 메모리, 입출력 능력을 통합해서 운영체제와 응용 프로그램 모두에게 단일한 가상 시스템을 제공한다.vSMP Foundation은 VFe나 vNUMA와 유사하게 단일한 실행 환경을 제공하기 위해 하이퍼바이저 형태의 소프트웨어 인터셉터 엔진을 사용한다.

vSMP Foundation은 노드 간 통신의 지연시간(latency)을 최소화하기 위해 best-of-breed와 같은 발전된 일관성 캐싱 알고리즘을 사용한다. 이 알고리즘은 실시간 메모리 사용 접근 패턴에 근거하여 퍼블록(per-block) 기반에서도 동시 동작한다.

이상과 같은 기능을 갖는 vSMP Foundation은 각 서버의 CPU와 메모리를 통합하여 확장 성능이 128노드, 최대 256TB 메모리의 단일 시스템 환경을 제공한다.

Ⅳ. 하드웨어 기반 원격 메모리 가상화

원격 메모리 가상화 기술 분야에서 지금까지 알아본 소프트웨어 접근 방법뿐 아니라, 오랜 시간 동안 클러스터 노드를 통합한 하드웨어 접근 방법도 지속적으로 연구되어 왔다. 이중 최근 가장 활발한 하드웨어 기반 원격 메모리 가상화 연구 개발을 진행 중인 SGI사의 Altix UV[6]와 NumaScale 사의 Numa-Connect[7]에 대해 설명 하겠다.

1. Altix UV

Altix UV는 SGI사에서 개발한 5세대 전역 공유 메모리 아키텍처로 하드웨어적으로 최대 16 테라바이트급 규모의 공유 메모리를 제공하는 단일 시스템 이미지 형태를 지원하는 플랫폼이다. Altix UV는 의사 결정 지원, 유전체학 및 생물 과학, 화학, 재료, 물리, 통합 시스템 과학, 국가 안보, 제품 설계 및 기타 데이터 집약적인 분야의 모든 영역에서 성능을 가속화 할 수 있는 플랫폼이다.

1.1. Altix UV의 구조 및 특징

Altix UV는 4096개의 프로세서 코어와 16 테라바이트 규모의 공유 메모리를 SGI의 차세대 고대역 및 저지연 연결망 기술인 NUMAlink 5를 통한 상호 연결한 구조로 되어 있다. 이러한 연결망 구조는 연구자가 동일한 작업을 동시에 더 많은 프로세서 코어에 할당할 수 있도록 높은 수준의 병렬 처리 효율성을 지원하기 위해 확장 가능한 공유 메모리 또는 메시지 전달을 특화시킨 형태를 지원한다.

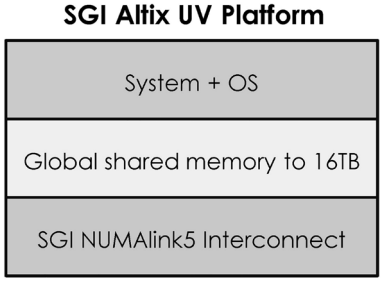

그림 8. SGI Altix UV 플랫폼 개념도

상용 클러스터는 개별 시스템이 독자적인 메모리와 운영체제를 갖고 InfiniBand와 같은 범용 연결망을 통해 통신한다. 노드 간 통신이 잠재적인 병목이 될 수 있고, 프로그래밍도 병렬 코드 수행이 요구될 수 있다. 이에 비해 SGI Altix UV는 모든 노드가 단일 공유 메모리 공간을 갖는다. 노드 간 데이터 전달이 감소하고, 대규모 데이터 전체가 단일 메모리 상에서 작업이 가능하며, 단일 노드 뷰의 프로그래밍으로 간단해지는 장점이 있다. SGI Altix UV의 주요 특징은 다음과 같다.

- 대규모 인-코어 컴퓨테이션
- 대규모 메모리 사상 입출력
- 고효율 응용 프로그램 확장과 메시지 전송
- 단순화된 응용 프로그램 부하 분산

1.2. 데이터 집중형 응용 최적화

단일 시스템 이미지로 최대 16 Tbytes의 글로벌 공유 메모리를 지원할 수 있어 인메모리 데이터베이스뿐만 아니라 다양한 형태의 데이터를 처리하는 응용 프로그램, 컴퓨팅 집중적인 HPC(High Performance Computing) 응용 프로그램의 운용에 효과를 볼 수 있다. 이런 처리능력은 클러스터 응용 프로그램들의 성능을 향상시키기 위해 대역폭을 양방향 15GB/s로 최대화하고 지연(latency)을 1μs이하로 최소화시키는 SGI

의 5세대 NUMAlink 연결망과 집적된 MPI(Message Passing Interface) 오프로드 엔진(MOE) 기술에 의한 것이다. SGI Altix UV의 고성능은 대용량의 스케일 아웃(scale-out) 클러스터와 결합해 분석용 슈퍼노드의 용도 및 독립적인 단일 형태 시스템의 용도로 사용하여 가장 복잡한 컴퓨팅 작업도 수행할 수 있게 한다.

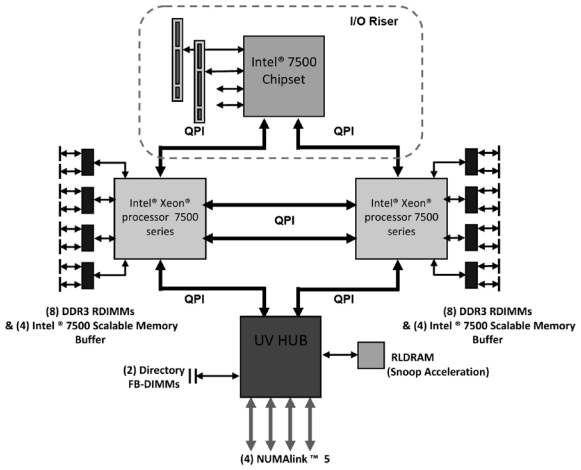

그림 9. Altix UV 컴퓨트 블레이드 블록도

SGI Altix UV 하드웨어 플랫폼은 '플러그 앤 솔브(plug and solve)'라는 구성 유연성을 제공하기 위한 모듈화 블레이드로 이뤄져 있다. NUMAflex 아키텍처는 적합한 시스템 규모의 구성을 가능하게 하고, 균형 잡힌 컴퓨팅과 메모리, 스토리지 구성을 할 수 있게 한다.

1.3. 개방형 하드웨어 플랫폼

Altix UV는 개방형 표준을 따른 하드웨어 플랫폼으로, 노벨의 수세 리눅스(SUSE Linux)나 레드햇 리눅스(Red Hat Linux)를 별도 수정 작업 없이 바로 사용할 수 있다. 경제적인 x86용 응용 프로그램을 수정 없이 사용할 수 있으며, 커스텀 코드 운용뿐 아니라 ISV(Independent Software Vendor) 응용 프로그램도 바로 운용할 수 있다. SGI는 사용자가 Altix UV 플랫폼의 성능을 최대화 할 수 HPC 솔루션 소프트웨어인 SGI 프로팩을 제공해 보다 빠르게 결과를 얻을 수 있도록 한다. 산업표준인 PCI 익스프레스 확장 슬롯의 채택으로 표준형 네트워킹 및 스토리지와 그래픽스 또는 GPU(Graphic Processing Unit) 카드 등을 사용하여 제한 없는 시스템 구성도 가능하다. 이런 모든 장점들을 적용하면 Altix UV가 공유메모리 방식뿐만 아니라 분산 메모리 응용 프로그램의 운용에도 고성능을 낼 수 있다.

2. NumaConnect

NumaScale의 NumaConnect는 하드웨어 칩과 연결망을 통해 캐쉬 일관성을 지원하는 공유 메모리 아키텍쳐이다. NumaConnect는 다수의 상용 및 범용 시스템을 연결하여 하나의 통합 OS에서 모든 프로세서, 메모리 및 입출력 자원을 공유하는 구조를 갖는다. NumaConnect기술은 클러스터 시스템의 비용 수준에서 기업의 메인 프레임의 기능과 확장 가능한 서버를 구축하는 것을 목표로 한다. 이 기술은 윈도우, 리눅스나 유닉스를 같은 표준 운영 시스템에 의해 제어되는 완전 가상화 환경에서 시스템의 모든 프로세서, 메모리 및 입출력 자원을 결합한다.

NumaConnect의 핵심은 NumaChip이다. 이 칩은 캐시 일관성 공유 메모리 제어 로직과 7-way 스위치를 결합한 단일 칩이다. NumaChip은 응용 프로그램이 클러스터 컴퓨팅을 위한 별도의 프로그래밍이 없이도 멀티코어 환경에서 원활하게 확장 될 수 있도록 지원한다. NumaConnect는 시스템의 모든 리소스에 대한 통일된 접근성 제공과 낮은 접근 지연을 위한 캐싱 기술의 활용에 의해 다른 연결망 장치와 차별된다.

2.1. NumaConnect의 구조 및 특징

클러스터 시스템에서 프로세스는 느슨한 결합 형태로 이더넷이나 InfiniBand 같은 범용 네트워크를 통해 연결된다. 단일 서버에 존재하는 자원보다 더 많은 프로세서나 입출력 장치를 활용해야하는 응용 프로그램은 처음부터 이것을 고려한 프로그래밍이 요구된다.

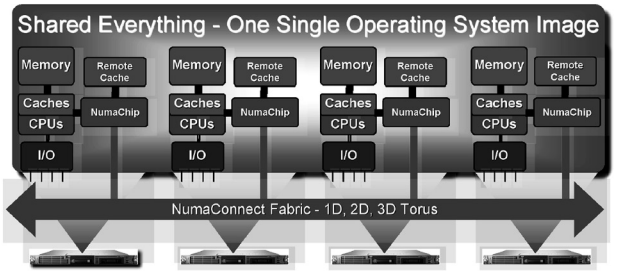

그림 10. NumaChip 시스템 아키텍쳐

NumaConnect는 공유 메모리의 위치 정보를 저장하는 분산 디렉토리에 기반 해서 모든 메모리를 공유 할 수 있는 확장 가능한 접근 방식을 사용한다. 이는 응용 프로그램을 변경하지 않고, 단일 시스템 보드의 한계를 넘어 확장 할 수 있다는 것을 의미한다. 시스템의 모든 프로세서 상에서 실행되는 프로세스는 메모리의 물리적 위치가 다른 시스템 보드에 있는지 여부에 관계없이 메모리를 사용할 수 있다. NumaConnect의 주요 특징

은 다음과 같다.

- AMD Opteron에 대한 확장, 디렉토리 기반 캐쉬 일관성 공유 메모리(ccNUMA) 연결망
- HTX(HyperTransport) 커넥터를 통해 하이퍼 트랜스 포트을 응집성 픽업 모듈 또는 시스템 보드에 직접 장착 가능
- 노드 별 설정 가능한 원격 캐시
- 전체 48비트 물리주소 공간(256 Tbytes)
- 최대 4,096 노드 지원
- 1μs 이내 MPI 응답 지연
- On-chip 2차원 또는 3차원 토로스 토폴로지 지원형 분산 스위치 패브릭

NumaChip을 통해 상호 연결된 시스템에서는, 모든 프로세서는 메인 프레임과 동일한 방식으로 시스템 내의 모든 메모리 및 모든 입출력 자원에 접근할 수 있다. NumaChip은 컴파일러로 생성된 병렬 프로세스와 쓰레드를 활용하는 메인프레임과 동일한 형태로 공유되는 메모리와 입출력 장치와 완전 가상화된 환경을 제공할 수 있다.

2.2. 캐시 일관성 지원 공유 메모리

다른 고속 연결망 기술과 비교해서 NumaConnect의 큰 차이점은 공유 메모리와 캐시 일관성 메커니즘에 있다. 이러한 특징들은 응용 프로그램에 대해 다중 프로세서 시스템에서 높은 수준의 효율성을 갖는 임의의 메모리 위치 및 메모리 사상 입출력 장치의 접근이 가능하게 한다. 그것은 수천 개의 프로세서로 구성된 단일 이미지 시스템이 노트북과 데스크톱에 사용되는 작은 다중 코어 시스템과 동일한 통합 프로그래밍 모델을 갖는 확장 가능한 시스템을 제공할 수 있게 한다.

클러스터 시스템과 비교해 볼 때, 공유 메모리 시스템은 다음과 같은 여러 가지 장점을 갖는다.

- 코딩 및 디버깅이 용이한 프로그래밍 편이성
- 상대적으로 적은 코드 규모
- 컴파일러는 루프 수준의 병렬 처리를 자동으로 이용 가능
- 적은 유지보수 노력으로 시스템 관리 가능
- 가상화 환경에서 자원의 최적 사용을 위해 시스템의 모든 프로세서에 자원이 사상될 수 있음
- 프로세스 스케줄링은 단일, 실시간 클럭의 동기화를 통해 낮은 효율의 스케줄링 직렬화를 피함

2.3. 확장성 및 견고성

NumaConnect의 설계는 16 비트 노드 식별자와 및 각 노드 내에서 48 비트의 주소로 이루어진 64 비트 물리적 주소 공간을 갖는 대규모의 프로세서 확장성이 목적이었다. AMD Opteron의 현재 구현은 4096 대의 물리 노드 표현에 사용되는 12 비트와 48 비트의 전역 물리 주소 공간에 의해 총 물리 주소 공간이 전체 256 Tbytes로 제약된다.

디렉토리 기반 캐시 일관성 프로토콜은 실제 데이터 처리량을 심각하게 감소시킬 수 있는 노드들 간의 일관성 트래픽에 의한 과부하를 방지하기 위해 데이터를 공유하는 노드들의 확장성을 처리하도록 개발되었다.

분산 스위칭의 기본 환형 토폴로지는 대부분의 다른 연결망보다 확장성 있는 다수의 다른 연결망 설정이 가능하게 한다. 이는 집중화된 스위치에 대한 필요성과 다차원 토폴로지를 위해 내재된 중복성을 제거한다.

V. 결 론

지금까지 원격 메모리 가상화 기술에 대해 소프트웨어 기반 방식과 하드웨어 기반 방식을 중심으로 몇 가지 사례와 기술적 동향을 살펴보았다.

요약해 보면 이들 원격 메모리 가상화 서비스 기술들은 본질적으로 고성능 컴퓨팅을 위한 분산 또는 클러스터 시스템의 통합을 위한 시도록 볼 수 있다. 각 시스템 기술들은 구현 방식에 따라 네 가지로 유형으로 분류될 수 있었다. 그러나 이 분류법은 최근 들어 더욱 구분하기 모호한 경우가 흔히 존재한다. 특정 구현 방식이 또 다른 구현 방식의 단점을 보안하는 경우도 많아 본고에서 설명한 몇 가지 시스템 구현 방법으로 원격 메모리 가상화 시스템을 한정하거나 획일화하기 보다는 서로 보완적인 다른 구현 방식을 상호 설계에 반영하는 것이 더욱 바람직해 보인다. 덧붙여, 대규모 데이터가 홍수처럼 넘치는 최근의 ICT 환경에서 대규모 메모리 기반 처리 모델과 분산 컴퓨팅 처리 환경을 충실히 활용함으로써 초다시점이나 UHD급 실시간 미디어와 같은 대규모 데이터 처리를 위해 보다 효율적인 컴퓨팅 인프라의 기반을 마련하는 것이 중요하다고 판단된다.

Acknowledgement

본 연구는 미래창조과학부 '범부처 Giga KOREA 사업'의 일환으로 수행하였음. [GK13P0100, Giga Media 기반 Tele-Experience 서비스 SW플랫폼 기술 개발]

참 고 문 헌

[1] P. Keleher, S. Dwarkadas, A. L. Cox, and W. Zwaenepoel. Treadmarks: Distributed shared memory on standard workstations and operating systems. In Proc. of the Winter 1994 USENIX Conference, pp. 115-131, 1994.

[2] Brett D. Fleisch and Gerald J. Popek. Mirage: A coherent distributed shared memory design. In Proceedings of the 12th ACM Symposium on OS Principles, pp. 211-223, 1989.

[3] Alex Vasilevsky. Linux virtualization on Virtual Iron VFe. In 2005 Ottawa Linux Symposium, July 2005.

[4] M. Chapman and G. Heiser. Implementing transparent shared memory on clusters using virtual machines. In Proc. of USENIX Annual Technical Conference, 2005.

[5] The Versatile SMP (vSMP) architecture and solutions based on vSMP Foundation. ScaleMP White Paper

[6] Technical Advances in the SGI Altix UV Architecture. SGI White Paper, 2009

[7] Einar Rustad. NumaConnect: A high level technical overview of the NumaConnect technology and products. NumaScale White Paper

[8] J.S. Robin and C.E. Irvine. Analysis of the Intel Pentium's Ability to Support a Secure Virtual Machine Monitor. In Proceedings of the 9th USENIX Security Symposium, pp. 129-144, August 20, 2000.

[9] R.P. Goldberg. Architectural Principles for Virtual Computer Systems. Ph.D. Thesis, Harvard University, Cambridge, MA, 1972.

[10] A. Whitaker, M. Shaw, and S. Gribble. Scale and Performance in the Denali Isolation Kernel. In ACM SIGOPS Operating system Rev., vol. 36, no SI, pp. 195-209, Winter 2000.

[11] InfiniBand Trade Association. Infiniband technology overview. http://www.infinibandta.org/about/, accessed on 30th September 2008.

[12] Intel Corp. Intel 64 and IA-32 Architecture Software Developer's Manual, August 2007. http://www.intel.com/products/processor/manuals/index.htm.

[13] John B. Carter. Design of the Munin distributed shared memory system. Journal of Parallel and Distributed Computing, 29: pp. 219-227, 1995.

[14] Brian N. Bershad and Matthew J. Zekauskas. Midway: Shared memory parallel programming with entry consistency for distributed memory multiprocessors. Technical Report CMU-CS-91-170, Carnegie Mellon University, 1991.

[15] B. Nitzberg and V. Lo. Distributed shared memory: A survey of issues and algorithms. IEEE Computer, 24(8):pp. 52-60, August 1991.

[16] P. Keleher, A. L. Cox, and W. Zwaenepoel. Lazy release consistency for software distributed shared memory. In Proceedings of the 19th Annual International Symposium on Computer Architecture, pp. 13-21, May 1992.

[17] K. Gharachorloo, D. Lenoski, J. Laudon, P. Gibbons, A. Gupta, and J. Hennessy. Memory consistency and event ordering in scalable shared-memory multiprocessors. In Proceedings of the 17th Annual International Symposium on Computer Architecture, pp. 15-26, May 1990.

[18] Intel Corp. Itanium Architecture Software Developer's Manual, October 2002. http://developer.intel.com/design/itanium/family/.

[19] E. Bugnion, S. Devine, M. Rosenblum. Disco: Running commodity operating systems on scalable multiprocessors. In Proc. 16th SOSP, pp. 27-37, 1997.

[20] K. Li, P. Hudak. Memory coherence in shared virtual memory systems. Trans. Comp. Syst., 7: pp. 321-59, 1989.

약 력

차 규 일

1998년 고려대학교 전산학과 이학사
2000년 고려대대학교 전산학과 이학석사
2000년~현재 한국전자통신연구원 클라우드컴퓨팅 연구부 선임연구원
관심분야: 운영체제, 원격메모리 가상화시스템

김 영 호

1999년 충북대학교 공학사
2001년 충북대학교 공학석사
2001년~현재 한국전자통신연구원 클라우드컴퓨팅 연구부 선임연구원
관심분야: 고성능컴퓨팅시스템, 분산 통합 메모리, NVRAM 하이브리드 메모리

안 신 영

1997년 성균관대학교 정보공학과 (학사)
1999년 성균관대학교 전기전자및컴퓨터공학부 (석사)
1999년~현재 한국전자통신연구원 클라우드컴퓨팅 연구부 선임연구원
관심분야: High Performance Computing, Software Architecture

임 은 지

1999년 부산대학교 이학사
2001년 부산대학교 이학석사
2011년~현재 한국전자통신연구원 클라우드컴퓨팅 연구부
관심분야: 분산시스템, 고성능컴퓨팅, 시스템소프트웨어, 운영체제