하이퍼쓰레딩을 위한 그리드 MDS의 설계 및 구현

이정훈° 오영은 김진석
서울시립대학교 컴퓨터과학부
{jhlee94, iceize, jskim}@venus.uos.ac.kr

Design and Implementation of GRID MDS for Hyperthreading

Jung-Hoon Lee°, Young-Eun Oh and Jin Suk Kim
School of Computer Science, University of Seoul

요 약

최근 많은 프로세서 제작업체들이 프로세서의 효율을 높이기 위한 방법으로 독립적인 쓰레드들을 한 프로세서 사이클에 동시에 실행시킬 수 있는 SMT 기술을 구현하고 있으며 그 예의 하나가 하이퍼쓰레딩이다. 물리 프로세서 안에 여러 개의 논리 프로세서를 가질 수 있는 하이퍼쓰레딩 기술은 응용단계에서 논리 프로세서들을 찾아내고 특정 논리 프로세서에 작업을 할당시킬 수 있는 방법이 필요하다. 따라서, 본 논문에서는 리눅스 운영체제에서 하이퍼쓰레딩 기술을 지원하는 마이크로프로세서의 특정 논리 프로세서를 탐지하고 제어하는 방안을 제시하고 이를 구현하였다. 또한 이를 그리드에 적용함으로써 그리드에서 하이퍼쓰레딩 기술을 지원하는 시스템을 올바르게 인식하고 적절하게 관리하여 효율적인 성능을 기대할 수 있게 되었다.

1. 서 론

프로세서의 자원 낭비를 줄이고 효율을 높이기 위해 1995년 Dean Tullsen에 의해 제안된 SMT(Simultaneous MultiThreading)[1] 기술은 최근 하이퍼쓰레딩(Hyper Threading)[2] 이라는 이름으로 실제 프로세서에 구현되었다. 하이퍼쓰레딩 기술은 한 개의 물리 프로세서가 두 개의 논리 프로세서를 가질 수 있게 함으로써 서로 다른 쓰레드를 동시에 처리할 수 있도록 설계되었다[3]. 하이퍼쓰레딩 기술은 각 논리 프로세서들이 하나의 물리 프로세서 안에서 Execution engine, 캐쉬, 시스템 버스 인터페이스 등과 같은 실행 자원들을 공유한다. 그러므로 효율적인 자원 활용과 프로세서 성능 향상을 위하여 기존의 일반적인 멀티 프로세서 환경과는 다른 방식으로 접근해야 할 필요가 있다. 또한 논리 프로세서는 고유의 Architectural state를 가지고 있기 때문에 응용 프로그램에서 정확히 논리 프로세서를 탐지하고 소속되어 있는 각 물리 프로세서를 밝혀내는 일은 중요한 이슈다. 따라서 본 논문에서는 리눅스에서 동작할 수 있도록 Cpucounter for Linux를 개발하여, 하이퍼쓰레딩 기술을 지원하는 마이크로프로세서의 특정 논리 프로세서를 탐지하고 제어하는 방안을 제시하고 이를 구현하였다. 또한 이를 전세계에 흩어진 이질적인 자원들을 이용하여 높은 성능을 이루고자 하는 그리드(GRID)[11]에 적용함으로써 그리드에서 하이퍼쓰레딩 기술을 지원하는 시스템을 올바르게 인식하고 논리 프로세서들을 적절하게 관리하여 효율적인 성능을 기대할 수 있게 되었다.

2. SMT 기술구현 동향

최근 인텔은 하이퍼쓰레딩이라는 이름으로 하나의 물리 프로세서가 두 개의 논리 프로세서로 동작하는 2-Context SMT 기술을 구현하였다. 각 논리 프로세서는 고유의 Architectural state를 가지고 다른 논리 프로세서와 독립적으로 동작할 수 있게 설계되어 각각 중지되거나 인터럽트를 받거나 또는 쓰레드를 처리할 수 있다[2]. 또한 Compaq에서는 알파칩에서 4-Context SMT 기술을 지원하는 8-way super-scalar out-of-order execution 프로세서인 Alpha 21464 (EV-8) 프로세서 개발 계획을 발표하였다. 그러나 HP와 합병되면서 EV-8 프로세서 개발계획은 무기한 연기된 상태이다[5]. 이 외에 Sun Microsystems[16], Memory Logix[17], Clearwater Networks[18]등에서 SMT 기술을 적용한 프로세서를 연구하고 있거나 또는 개발 중에 있다.

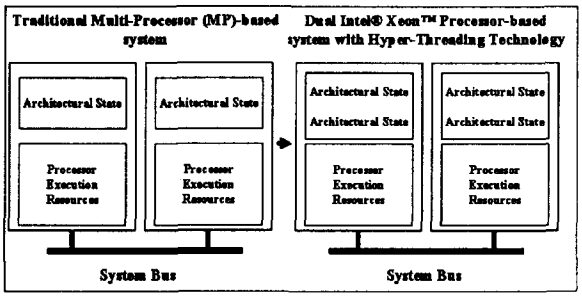

그림 1 인텔 하이퍼쓰레딩 기술[8]

이러한 SMT 기술을 구현한 프로세서의 성능향상을 위해서는 Cpucounter for Linux와 같은 특정 논리 프로세서를 탐지하고 제어하는 방안이 필요하다.

3. Cpucounter for Linux의 설계 및 구현

Cpucounter for Linux는 인텔의 윈도우즈 기반 cpu counting utility와 같은 설계로 구현되었다[6]. 하이퍼쓰레딩 기술이 지원되는 프로세서에서 논리 프로세서를 탐지하고 제어하기 위해서는 크게 다음과 같은 3가지 단계를 밟는다.

1) 프로세서의 하이퍼쓰레딩 기술 지원 여부 판단
2) 각 물리 프로세서에 포함된 논리 프로세서의 개수 판단
3) 논리 프로세서와 물리 프로세서의 연관관계 판단

3.1 프로세서의 하이퍼쓰레딩 기술 지원 여부 판단

프로세서가 하이퍼쓰레딩 기술을 지원하는지의 여부를 판단하기 위해 인텔 프로세서에서 모델 정보를 출력하는 cpuid 인스트럭션을 사용할 수 있다[7]. 프로세서의 EAX 레지스터 값을 0H로 할당하고 cpuid 인스트럭션을 실행하면 인텔 프로세서의 벤더 정보가 출력된다. 우선 이 벤더정보를 토대로 하여 인텔의 프로세서인지 AMD 등의 다른 x86 호환 프로세서인지 판독해야 한다. 현재까지 하이퍼쓰레딩 기술은 인텔 프로세서에서만 지원된다. 벤더 정보를 토대로 EAX 레지스터 값을 1H로 할당하고 cpuid 인스트럭션을 실행하면, EDX 레지스터의 28번째 비트로부터 프로세서의 하이퍼쓰레딩 지원 여부를 알수 있다[9]. 이 값은 프로세서가 하이퍼쓰레딩 기능을 수행하고 있는지에 대한 동적인 정보를 나타내는 것이 아니며 하이퍼쓰레딩 기능을 제공할 수 있는지에 대한 정적인 상태정보를 나타낸다.

3.2 각 물리 프로세서에 포함된 논리 프로세서의 개수 판단

물리 프로세서당 논리 프로세서 개수를 판단하기 위하여 마찬가지로 cpuid 인스트럭션을 사용한다. EAX 레지스터 값을 1H로 할당하고 cpuid 인스트럭션을 실행하면 EBX 레지스터 16-23비트에 물리 프로세서당 논리 프로세서의 개수 정보가 출력된다[9]. 기본적으로 하이퍼쓰레딩 기능을 지원하지 않는 인텔 프로세서에서는 논리 프로세서의 개수가 1이다.

3.3 논리 프로세서와 물리 프로세서의 연관관계 판단

프로세서에서 하이퍼쓰레딩 기능을 제공할지라도 BIOS, 운영체제, 응용 프로그램에 의하여 하이퍼쓰레딩 기능을 제한하거나 식별하지 못하여 하이퍼쓰레딩 기능을 활용하지 못할 수 있다. 따라서 논리 프로세서와 물리 프로세서의 연관 관계를 판단하는 것은 시스템에서 하이퍼쓰레딩 기능을 제공하는지 여부를 최종적으로 판단하는 데 중요하다. 시스템에서는 각 장치들을 구분하기 위하여 APIC (Advanced Programmable Interrupt Controller) ID를 사용한다. BIOS는 각 논리 프로세서마다 고유의 APIC ID를 할당하며, 이 APIC ID를 통하여 각각의 논리 프로세서를 제어할 수 있다.

		31	23	15	7	0
Misc. Info	EBX	APIC ID	Count	Chunks	Brand ID	
Feature Flags	ECX	Bit Array				
	EDX	Bit Array				

그림 2. EAX=1H 일 때 cpuid 인스트럭션 결과[9]

<그림 2>에서 보는 바와 같이 EAX 레지스터에 1H 값을 할당하여 cpuid 인스트럭션을 실행시키면, EBX 레지스터의 24-31 비트값에 프로세서의 APIC ID가 출력된다. 논리 프로세서의 APIC ID는 <그림 3>과 같이 해당 논리 프로세서가 포함되어 있는 물리 프로세서의 APIC ID와 자신의 식별자로 구성되어 있다.

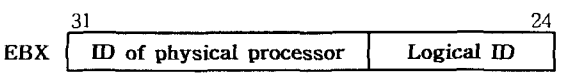

그림 3. 하이퍼쓰레딩 프로세서의 APIC ID 구조

현재까지 하이퍼쓰레딩 기술은 2개의 논리 프로세서를 지원하므로 24번째 비트값이 논리 프로세서의 구분자 역할을 담당하며, 따라서 같은 물리 프로세서에 포함된 논리 프로세서들은 25-31 비트값이 같다. 해당 프로세서의 APIC ID를 출력하기 위하여 응용 프로그램에서는 현재 운영체제에서 활용 가능한 프로세서 중에서 APIC ID를 추출하기 위한 특정 프로세서를 대상으로 선택적으로 명령을 실행시킬 수 있어야 한다. 이를 위해서 system affinity 값과 process affinity 값, 그리고 각각의 mask 값을 사용한다. system affinity 값은 운영체제에서 활용할 수 있는 프로세서의 개수를 나타내며, process affinity 값은 하이퍼쓰레딩과 같은 멀티 프로세서 환경에서 해당 프로세스가 어느 프로세서에서 작업을 수행할 것인지를 결정한다. 또한 system affinity mask는 현재 운영체제에서 사용하는 프로세서의 개수를 결정하며, process affinity mask는 프로세스가 선택적으로 실행될 수 있는 프로세서의 값을 결정한다. affinity를 설정하거나 추출하기 위해서는 운영체제 차원에서 적절한 시스템 콜을 제공해 주어야 하며, 이것은 리눅스 커널 2.4 버전부터 부분적으로 지원된다[10]. 즉, 2.5.8-pre3 미만의 버전을 가진 리눅스 커널에서는 affinity 관련 시스템 콜을 동작시키기 위하여 적절한 패치를 해야 하며, 그 이상의 버전에서는 sched_setaffinity와 sched_getaffinity 시스템 콜을 이용하여 특정 논리 프로세서를 대상으로 APIC ID를 추출할 수 있다. 최종적으로 논리 프로세서와 물리 프로세서의 연관관계는 현재 운영체제에서 파악하고 있는 프로세서의 개수와 APIC ID를 비교하여 종합적으로 판단해야 한다. 운영체제의 관점에서 논리 프로세서는 독립적인 하나의 프로세서로 보여지므로 프로세서 정보를 얻어내기 위하여 proc 파일 시스템에 존재하는 상태 정보를 추출한다. 이렇게 얻어진 프로세서 정보와 각 프로세서별로 실행시킨 APIC ID와 대조하여 실제 물리 프로세서 개수와 논리 프로세서 개수, 각 논리 프로세서가 속한 물리 프로세서 등을 종합적으로 파악할 수 있다.

3.4 그리드 자원 관리 지원

지역적으로 분산되어 있는 이질적인 자원들을 하나로 묶어 거대한 시스템을 구성하고자 하는 연구가 진행되고 있으며, 그 예의 하나가 그리드(GRID)[11]이다. 이러한 시스템을 구성하는 자원들은 특성이 다르기 때문에 같은 작업을 실행할 때 실행 능력이 다르게 된다[12]. 그리드 시스템이 효율적으로 운용되기 위해서는 전세계에 분산되어 있는 이질적인 자원 정보를 관리하고 모니터링을 담당하는 자원 정보 관리자가 시스템들의 자원 정보를 올바르게 파악해야 한다. 한편 하이퍼쓰레딩 기술을 지원하는 마이크로프로세서는 일반적인 멀티프로세싱 환경과 다르기 때문에 프로세서 자원을 효과적으로 활용하기 위하여 다른 관리방법을 택해야 한다. 예를 들어 그리드 자원관리자가 2-context SMT Dual 시스템을 4-way SMP (QUAD) 시스템으로 판단하고 작업을 스케줄링한다면 효율적인 시스템 성능을 기대하기 어렵다. 따라서 그리드에 해당 시스템의 하이퍼쓰레딩 관련 정보를 올바르게 제공한다면 하이퍼쓰레딩 시스템뿐만 아니라 그리드 전체의 성능을 높여주게 된다. 본 논문에서는 Cpucounter for Linux를 이용하여 그리드 자원 관리자인 MDS(Monitoring and Discovering Service)[14]에 적절한 패치를 함으로써 하이퍼쓰레딩 시스템을 그리드에서 인식하고 효율적으로 관리할 수 있도록 하였다. 또한 이를 그리드 이용자가 편리하게 그리드 자원을 이용할 수 있게 설계된 웹기반 인터페이스인 SWING(Simple Web Interface for Network of

Interface)[15]과 연동하여 그리드에서 하이퍼쓰레딩 시스템 자원관리 여부를 쉽게 파악할 수 있도록 하였다. 하이퍼쓰레딩 기술을 지원하는 프로세서 자원을 MDS에서 관리하기 위하여 데이터 모델에 맞추어 MDS 자원관리 속성항목에 <표 1>과 같은 6개 항목을 추가하였다.

속 성	설 명
Mds-Cpu-Smt-vendor	프로세서 제조업체 정보
Mds-Cpu-Smt-brand	프로세서 모델명
Mds-Cpu-Smt-capable	하이퍼쓰레딩 지원 가능여부
Mds-Cpu-Smt-Physical-count	실지 물리 프로세서 개수
Mds-Cpu-Smt-LP-Count	물리 프로세서당 논리 프로세서 개수
Mds-Cpu-Smt-enabled	CPU, BIOS, 운영체제, 응용 프로그램 모두 하이퍼쓰레딩 지원가능 여부

표 1. MDS 패치후 추가되는 속성정보값

<표 1>의 새롭게 추가된 항목중 Mds-Cpu-Smt-enabled 항목은 현재 운용되고 있는 시스템의 프로세서가 하이퍼쓰레딩 기능을 지원하더라도 BIOS, 운영체제, 또는 응용 프로그램에서 하이퍼쓰레딩 기능을 인식하지 못하거나 또는 동작하지 않게 함으로서 하이퍼쓰레딩 기능이 지원되지 않는 상황을 나타낸다. 현재는 Globus 2.0 버전을 기준으로 MDS 패치가 이루어졌으며, 이를 통하여 그리드에서 하이퍼쓰레딩 시스템을 인식하고 올바르게 작업을 할당할 수 있는 기반이 마련되었다. <그림 4>는 SWING에 Cpucounter for Linux 툴킷의 패치를 가한 모습으로서 그리드에서 하이퍼쓰레딩 시스템을 인식하고 있음을 보여주고 있다.

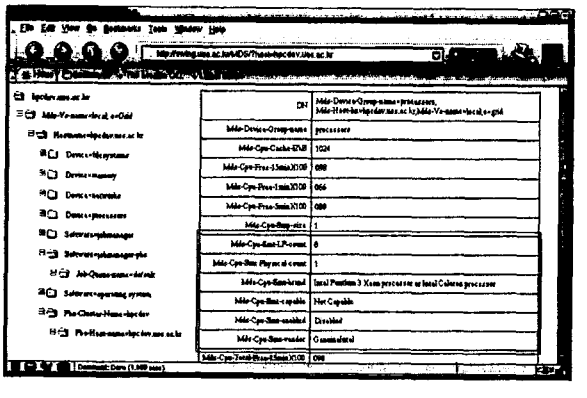

그림 4. SWING 의 하이퍼쓰레딩 시스템 자원관리

4. 결론

본 논문에서는 리눅스 운영체제에서 하이퍼쓰레딩 기술을 지원하는 마이크로프로세서의 특정 논리 프로세서를 탐지하고 각각의 논리 프로세서를 제어하는 방안을 제시하고 구현을 통해 보였다. 또한 이를 그리드에 적용함으로써 그리드에서 하이퍼쓰레딩 기술을 지원하는 시스템을 올바르게 인식하고 관리하여 효율적인 성능을 기대할 수 있게 되었다. 향후에는 리눅스 커널에서 SMT 프로세서를 효과적으로 스케줄링하고, 다중 Context SMT 환경에서 성능향상을 위한 효율적인 논리 프로세서 제어 방안과 그리드에서 SMT 프로세서 자원들을 적합하게 관리할 수 있는 방안에 대한 연구가 진행되어야 할 것이다.

참고문헌

[1] D. Tullsen. S. Eggers, J. Emer, H. Levy, J. Lo, and R. Stamm, "Simultaneous Multithreading: Maximizing On-Chip Parallelism," *Proc. of the 22rd Annual International Symposium on Computer Architecture*, June, 1995.

[2] Intel Corporation., "Introduction To Hyper-Threading Technology," Document number 250008-002, 2001.

[3] Deborah T. Marr et al, "Hyper-Threading Technology Architecture and Microarchitecture," *Intel Technology Journal*, Q1, 2002.

[4] Cpucounting utility, http://cedar.intel.com/cgi-bin/ids.dll/content/content.jsp?cntKey=Generic+Editorial%3a%3acodesample_cpucounter_desc&cntType=IDS_EDITORIAL&catCode=CZC

[5] Compaq memorial site, http://www3.sympatico.ca/n.rieck/links/compaq_memorial_site.html

[6] Intel corporation, "Detecting Support for Hyper-Threading Technology Enabled Processors," Dec, 2001.

[7] Intel corporation, "IA-23 Intel Architecture Software Developer's Manual, Volume 1: Basic Architecture," Order number 245472, 2001.

[8] Intel corporation, "White Paper: Hyper-Threading Technology on the Intel Xeon Processor Family for Servers," 2002.

[9] Intel corporation, "Intel Processor Identification and the CPUID Instruction," Order Number 241618-022, 2002.

[10] Cpu affinity kernel patch, http://www.kernel.org/pub/linux/kernel/people/rml/cpu-affinity

[11] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *Journal of High-Performance Computing Applications*, vol. 15, no. 3, pp. 200-222, 2001.

[12] 김학두, 김진석, "이질적인 계산자원환경에서 독립적인 작업들을 위한 온라인 휴리스틱 스케줄링 알고리즘," *정보과학회 '2002 추계 학술발표 논문집*, 제 29권 제 2호., pp. 304-306, 2002.

[13] Globus Toolkit, http://www.globus.org.

[14] The Monitoring and Discovery Service, http://www.globus.org/mds

[15] 오영은, 김진석, "SWING: Globus를 위한 웹 인터페이스의 설계 및 구현," *정보과학회 '2002 추계 학술발표 논문집*, 제 29권, 제 2호, pp. 325-327, 2002.

[16] EETimes, http://www.eetimes.com/story/OEG20011210S0069

[17] Memory Logix, "MLX1 A tiny multithreaded 586 core for smart mobile devices," http://www.cs.washington.edu/research/smt/memoryLogix.pdf

[18] Clearwater Networks, http://www.clearwaternetworks.com/splash.html