매니코어 구조에서의 효율적 자원 할당을 위한 Cilk 확장성 모델

송욱, 김주성, 김지홍
서울대학교 컴퓨터공학부
e-mail : wooksong@davinci.snu.ac.kr

A Cilk Scalability Model for Efficient Resource Allocation on Manycore Architectures

Wook Song, Joosung Kim, Jihong Kim
*School of Computer Science and Engineering, Seoul National University

요 약

매니코어 시스템에서는 프로그램의 확장성에 대한 정보가 코어와 같은 병렬 자원의 할당 문제 해결에 핵심적인 역할을 한다. 본 논문에서는 Cilk 런타임 시스템에서 구동되는 응용 프로그램들에 대한 확장성 모델을 제안하여 매니코어 시스템에서의 효율적인 자원 관리에 활용하고자 한다. 특히, 네트워크-온-칩 구조 및 디렉터리 기반 캐시 일관성 프로토콜을 감안한 지연 시간 모델링을 통해 보다 정확한 성능 변화의 경향을 예측하고자 하였다. 최대 36 개까지의 코어 할당에 대한 지연 시간 예측 실험에서, 제안된 모델은 13%의 평균 오차를 보였다.

1. 서론

마이크로 프로세서는 설계의 복잡도 상승, 고집적화로 인한 높은 소비 전력과 발열, 물리적 한계로 인한 제조공정 미세화의 어려움 등으로 인하여 단일 프로세서 구조에서 멀티 프로세서 구조로 이행하여 왔다. 이러한 여러 개의 프로세서의 집적을 통하여 성능 한계를 극복하려는 최근의 추세는 수십 개에서 수천 개의 프로세서를 하나의 칩으로 집적한 매니코어 프로세서의 개념을 등장시켰다.

기존의 멀티코어 프로세서에 비해 비약적으로 많아진 코어의 수로 인하여 매니코어 프로세서는 코어 등의 병렬 자원의 관리 및 활용이 매우 어렵고 복잡하다. 따라서 매니코어 프로세서가 제공하는 높은 병렬성을 효율적으로 활용하기 위해서는 하드웨어 및 시스템 소프트웨어 수준의 자원 관리 계층이 필요하다. 이러한 자원 관리 계층에서는 동시에 각각의 코어를 점유하여 수행되는 프로그램 집단에 대해 시스템 전체적인 관점에서의 성능-에너지 효율이 최적인 선택이 요구되며, 이에 따르는 자원 할당 변동에 의해 발생하는 성능 및 소모 에너지 변화를 예측할 수 있어야 한다. 따라서, 이러한 정보를 제공하는 예측 모델의 정확성은 성능-에너지 최적 자원 할당의 결정의 문제뿐만 아니라, 매니코어 프로세서를 탑재한 시스템 전체의 효율적 활용에 있어 매우 중요한 문제라 할 수 있다.

한편, 매니코어 프로세서가 제공하는 매우 높은 병렬성을 효과적으로 사용하기 위한 다양한 프로그래밍 모델이 최근 제안되었다. Cilk[1], TBB (Threading Building Block)[2], X10[3]과 같은 태스크 기반 프로그래밍 모델은 프로그램을 작은 태스크로 나누어 이를 복수의 코어에 스케줄링 함으로써 기존의 쓰레드 기반 병렬 프로그래밍 모델보다 미세한 수준의 병렬화가 가능하게 하였으며, 사용하는 프로세서의 수에 대해 유연한 스케줄링을 지원한다는 장점을 가지고 있다.

본 연구에서는 태스크 단위의 런타임 시스템 기반 프로그래밍 모델인 Cilk 언어를 선택하여 Cilk 응용 프로그램에 할당될 수 있는 물리적 코어의 수와 할당된 코어들의 위치 등 특정한 조건에서의 응용 프로그램 성능 예측이 가능한 확장성 모델을 제안한다. 제안된 모델은 프로그램의 이전 수행 과정에서 수집된 수행시간, 특정 하드웨어 이벤트 횟수 등의 정보를 바탕으로 매니코어 프로세서에서 특징적으로 나타나는 하드웨어 구성요소인 네트워크-온-칩 구조(Network-on-Chip, NoC)에 의한 성능 지연 모델링을 통해 예측의 정확도를 향상시키고자 하였다.

본 논문의 구성은 다음과 같다. 2 장에서는 관련 연구를 소개하고, 3 장에서는 본 연구의 목표 및 기여를 서술한다. 4 장에서는 제안한 모델에서 가정하는 매니코어 프로세서의 구조 및 기반 시뮬레이션 환경을 설명한다. 5 장에서는 본 논문에서 제안하는 매니코어를 위한 확장성 모델을 소개하고 6 장에서는 5 장에서 제안된 확장성 모델의 각 요소의 정확성을 실험 결과를 통하여 검증한다. 마지막으로 7 장에서

는 본 연구의 결론을 내린다.

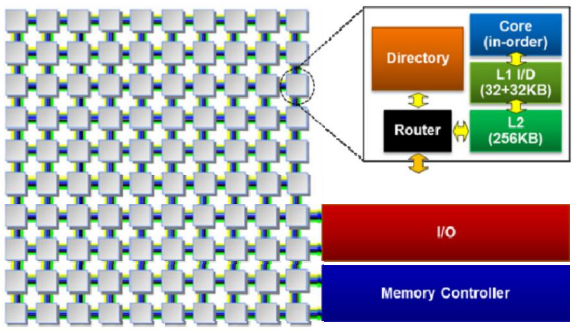
2. 관련 연구

병렬 프로그램의 확장성 연구는 쓰레드 모델의 등장과 함께 꾸준히 연구되고 있는 분야이며, 다양한 병렬 프로그래밍 모델과 분산 시스템 등을 기반으로 한 연구가 진행되고 있다. Cilkview[4] 에서는 Cilk 응용 프로그램의 태스크 이동 부하를 고려한 총 작업량 및 최상 경로 정보를 활용하여 멀티코어 시스템에서의 확장성 모델을 고안하였다.

3. 연구의 목표 및 기여

Cilk 런타임 시스템은 유휴 상태의 코어가 현재 태스크를 수행 중인 코어에 대기하고 있는 태스크를 가져오는 work-stealing 기법을 사용한다. 이러한 특성에 따라, Cilk 응용 프로그램은 타 병렬 프로그래밍 언어들과 달리 태스크의 이동이 발생할 경우에만 주로 추가적인 부하가 발생한다. Cilkview 에서 제안하는 예측 모델은 이러한 특성에 따라 특정한 Cilk 응용 프로그램에서 수행되는 전체 작업량과 최상 경로의 길이가 변하지 않는다는 가정 하에 코어 수 변화에 따른 수행 시간을 예측한다. 그러나 이러한 가정은 비교적 적은 수의 코어와 공유 버스로 구성된 SMP(Symmetric MultiProcessing) 구조에서만 성립되는 것으로, 수십 개 이상의 코어를 가진 매니코어 구조에서는 트래픽 문제로 인하여 공유 버스와 같은 연결 구조를 사용하기에 적합하지 않으므로 이와는 다른 경향을 보인다. 즉, 코어의 수뿐만 아니라 각 태스크가 수행되는 코어의 위치, 코어 간의 거리 등이 수행 시간 예측에 함께 고려되어야 한다.

본 논문에서는 매니코어 구조에서 특징적으로 나타나는 NoC 구조 및 디렉터리 기반 캐시 메모리 일관성 프로토콜을 감안하여 기존의 Cilk 성능 예측 모델을 확장한다. 제안한 모델을 통해, 매니코어의 자원 할당 문제에서 시스템 전체 성능을 고려한 효율적인 선택에 도움이 될 수 있는 성능 변화 정보를 제공하는 것이 목표이다.



(그림 1) 목적 매니코어 프로세서의 블록 다이어그램

4. 연구의 동기 및 연구 환경

4.1 목적 매니코어 프로세서의 구조 및 연구 환경

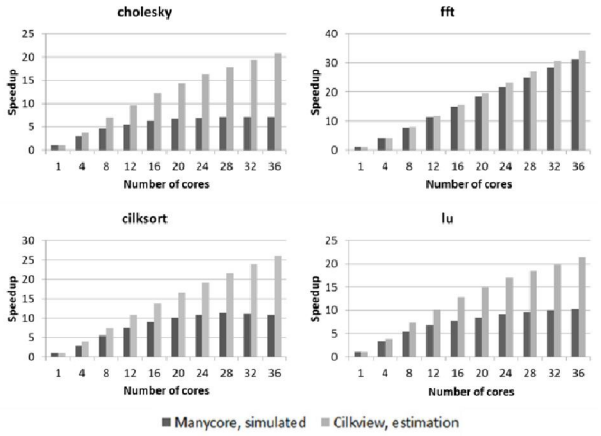
그림 1 은 본 연구에서 가정하는 매니코어 프로세서의 구조로써 64 개의 코어 타일이 NoC 구조로 연결된 것이다. 각 코어는 타일에 포함 된 라우터를 통해 메인 메모리, I/O 채널 및 다른 코어 타일 등에 접근한다. 이 밖에 각 타일이 포함하고 있는 디렉터리는 각 코어의 L2 캐시 메모리가 공유하는 데이터 간의 일관성을 유지하기 위한 것이다. 각 타일은 패킷 스위칭 방식의 이차원 mesh interconnection network 를 통해 연결된다.

본 논문에서 사용된 McSim[5] 시뮬레이터는 event-driven 방식의 매니코어 시뮬레이터로, 타이밍 시뮬레이션과 기능 시뮬레이션이 분리되어 있다. 기능에 대한 시뮬레이션은 Pin binary instrumentation [6] 도구와 이에 포함된 사용자 수준 pthread 라이브러리를 사용한다. McSim 시뮬레이터는 Pin 의 사용자 수준 pthread 라이브러리를 통해 재구성된 프로그램의 명령어 단위 정보를 수집하여 이를 내부의 가상 프로세서에서 동작시키는 방식으로 타이밍을 계산한다.

Cilk 런타임 라이브러리 및 Cilk 응용 프로그램이 시뮬레이터 환경에서 구동되도록 하기 위해서는 Pin 의 사용자 수준 pthread 라이브러리로 다시 컴파일 될 필요가 있으며, 이 밖에도 Cilk 내에서 코어 간 태스크 이동 및 동기화의 부하를 감소시키기 위해 사용하는 memory barrier 등의 시스템 종속적인 부분에 대한 수정이 필요하였다.

4.2 자원 할당에 따른 성능 변화

Cilk 와 같이 할당되는 코어의 수에 대해 능동적으로 작업량의 분배가 이뤄지는 프로그래밍 모델의 경우, 할당 코어의 수에 따라 프로그램의 수행시간이 변화한다. Cilkview 등에서는 Cilk 프로그램의 전체 작업량, 최상 경로 및 태스크의 이동 부하를 고려하여 코어 할당 수에 따른 성능 변화를 모델링하였다. 그러나 해당 모델은 CMP 구조를 대상으로 하여, 매니코어 프로세서에서 특징적으로 나타나는 패킷 스위칭 방식의 NoC 구조는 데이터 통신이 이뤄지는 주체 사이의 거리, 위치, 그리고 같은 라우터를 거쳐가는 트래픽의 양 등에 따라 메모리 요청에 대한 대기 시간이 가변적이다. 그림 2 는 Cilkview 의 추정 결과 대비 매니코어 시뮬레이션에서의 추정 결과를 비교한 것으로, 높은 불일치가 나타남을 볼 수 있다. 이러한 경향은 표 1 에서의 NoC 에 의한 높은 지연 시간 비율을 볼 때 모델에서 이를 고려하지 않았기 때문임을 짐작할 수 있다.



(그림 2) Cilkview 모델과 시뮬레이션 결과 비교

<표 1> 36 코어 할당시 전체 수행시간에 대한 NoC 지연시간의 비율

Benchmark	cholesky	cilksort	fft	lu
NoC Delay %	44.19	32.14	22.63	56.86

이러한 경향에 따라 매니코어 프로세서에서의 자원 할당에 따른 응용 프로그램의 성능을 예측하기 위해서는 각 코어에서 발생하는 트래픽의 양 및 패킷의 종류에 따른 라우팅 경로의 길이 등을 분석하고 이를 체계화 하는 것이 중요한 문제이다.

5. 매니코어 자원 할당을 위한 확장성 모델

5.1 NoC 플릿 발생 수 모델

NoC 구조에서는 데이터 전송의 기본 단위로 플릿(flit)을 사용한다. 각 플릿의 크기는 일반적으로 메모리 체계의 상위 수준인 L2 에서 사용되는 패킷의 크기보다 작으며, 따라서 L2 와의 통신에서 데이터가 전달되는 경우에는 하나의 헤더 플릿과 복수의 데이터 플릿을 보내는 방식을 사용한다. 이에 비해 데이터가 포함되지 않은 데이터 요청, 일관성 유지 신호 등에 대해서는 각각 하나의 플릿을 사용하여 통신한다.

NoC 에서 발생하는 플릿의 수가 할당되는 코어의 수에 대해 비례하여 변화한다는 특성을 이용하여, Cilk 응용 프로그램에 할당되는 코어의 수에 대한 NoC 플릿 발생 수를 다음과 같이 모델링한다.

$$EventCount_P \approx P * (\sum_{i=0}^{N} \frac{EventCount_{Pi}}{P_i})/\mathrm{N} \quad (1)$$

특정한 Cilk 응용 프로그램에 대해, P 개의 코어를 할당하였을 때의 디렉터리 관련 하드웨어 이벤트의 수 EventCount 는 N 개의 샘플 데이터를 기반으로 평균 기울기를 사용하여 추정한다. 이렇게 추정된 각 디렉터리 관련 이벤트의 수를 적용하여 P 개의 코어 할당 시 플릿의 수를 추측할 수 있다.

5.2 디렉터리 별 라우팅 거리 모델

디렉터리 기반 캐시 일관성 프로토콜에서는 플릿의 전달이 L2 와 디렉터리 사이에서만 발생하며, 플릿 전달에서 두 지점 사이의 거리는 라우터 간 hop 수로 표현된다. 따라서 특정한 L2 타일에서의 플릿 라우팅 거리는 해당 타일과 각 디렉터리와의 hop 수를 기반으로 한다. 본 논문에서 가정하는 온-칩-네트워크는 XY 라우팅을 사용하므로, 특정한 두 지점 A(xA, yA)와 B(xB, yB) 사이의 라우팅 거리는 다음과 같이 계산될 수 있다.

$$Distance(A,B) = |x_A - y_B| + |x_A - y_B| \quad (2)$$

정확한 전체 NoC 지연 시간을 계산하기 위해서는 각 태스크에서 발생하는 각 플릿의 라우팅 거리를 미리 알고 이를 개별적으로 계산할 필요가 있다. 본 모델에서는 해당 Cilk 응용 프로그램에서 발생하는 각 디렉터리에 대한 플릿 통신 비율에 기반한 weighted distance 를 계산한다. . 각 디렉터리에 대한 플릿 통신의 수를 FlitRatio(Dir)이라고 할 때, 특정 코어 A 에서 발생하는 플릿들의 평균 라우팅 거리는 다음과 같이 계산될 수 있다.

$$\mathrm{Average}Distance(A) = \sum_{i=0}^{P}(FlitRatio(Dir_i) * Distance(A, Dir_i)) \quad (3)$$

5.3 NoC 지연 시간 모델

NoC 플릿의 특성을 고려한 NoC 의 지연시간 모델은 다음과 같다.

$$NoCDelay(A) = HopDelay * HopCount(A) \quad (4)$$

$$HopCount(A) = AverageDistance(A) * \frac{Request(P) + Reply(P) + Coherence(P)}{P} + \left(\frac{DataPacketSize}{FlitSize} - 1\right) * \frac{DataOnly(P)}{P} \quad (5)$$

해당 지연 시간 모델은 크게 헤더 플릿에 의한 hop 수와 데이터 플릿에 의한 hop 수의 두 부분으로 나눌 수 있다. 헤더 플릿으로 인한 hop 수는 해당 코어의 수행시간에 직접적으로 영향을 주는 부분으로, 플릿의 라우팅 거리에 따라 차이를 보인다. 여기에는 Request, Reply, Coherence 의 세 가지 종류의 헤더 플릿이 속한다. 이에 비해 데이터 플릿은 헤더 플릿과 함께 연속적으로 전송되는 플릿으로, 라우팅에 의한 지연 시간은 헤더 플릿의 라우팅 시간에 겹쳐지는 경향을 보인다. 따라서 각 데이터 플릿은 귀속되는 헤더 플릿의 지연 시간에 추가적인 1 hop 만큼의 지연 시간으로 나타내어 질 수 있다. 데이터 패킷의 크기는 온-칩-네트워크의 상위 메모리 수준인 L2 에서의 각 캐쉬 라인의 크기와 같다. 따라서 L2 캐쉬 라인의 크기에 해당하는 수의 플릿에서 헤더 플릿을 제외한 수가 각 데이터 패킷에 대한 플릿의 수라고 할 수 있다.

6. 실험 결과

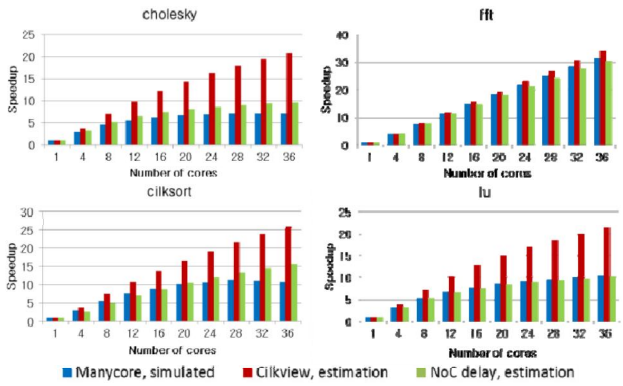
6.1 실험 환경

본 실험에서는 Pin 2.7 버전에서 구동되는 수정된 Pin 전용 pthread library/tool 1.0 및 McSim 1.0 을 사용하였으며 Cilk 5.4.6 을 McSim 을 지원하도록 수정하여 사용하였다. 우리는 고안한 확장성 모델의 정확성을 검증하기 위하여 시뮬레이터의 각 경우에 대한 NoC 지연시간과 모델의 예측 결과를 비교하였다. 이 과정에서 Cilk 로 제작된 SPLASH-2[12] 기반의 cholesky, fft, lu 외에도 병렬성이 높은 matmul, strassen 등의 벤치마크를 사용하였다. 표 2 는 McSim 에서 사용하는 가상 매니코어 프로세서의 파라미터를 정리한 것이다.

<표 2> 매니코어 프로세서의 파라미터

파라미터	값	
CPU	64-core, in-order	
Cache	private L1 I/D	32KB, 8-way, 2 cycle latency
	private unified L2	256KB, 4-way, 10 cycle latency, directory-based coherence
Interconnection Network	8x8 2d-mesh, XY routing, 10 cycle per-hop latency	
Memory	300 cycle latency	

6.2 NoC 지연시간 모델 검증 실험 결과



(그림 3) 확장성 모델의 예측 결과 비교

그림 3 은 1 개부터 36 개까지의 코어 수 할당에 대하여 NoC 에 의한 지연 시간의 예측 결과와 기존 모델인 Cilkview 에 의한 예측 결과, 시뮬레이터에 의해 시뮬레이션한 결과의 비교이다. 결과에서 볼 수 있듯이, NoC 를 경유하는 통신량이 높게 나타나는 응 용 프로그램들의 경우 기존 모델에서 제대로 예측하지 못하는 경향성을 수정해 줄 수 있는 역할을 수행할 수 있음을 볼 수 있다. 실험에서 사용한 벤치마크들에 대해, 확장성 예측 결과는 일반적으로 13%의 평균 오차를 보였으며, 벤치마크에 따라 최소 평균 오차는 3.54%, 드물게 병렬성이 떨어지는 경우에 대해 38.76%의 최대 평균 오차를 보였다.

7. 결론

매니코어 기반 시스템에서의 효율적인 코어 자원의 활용을 위해, 코어 자원 할당에 따른 프로그램의 성능을 예측하는 것은 중요한 문제이다. 한편, Cilk 와 같이 변화하는 코어의 수에 능동적으로 대응할 수 있는 프로그래밍 모델은 매니코어 기반 시스템에서도 잘 적용될 수 있을 것이라고 예측된다.

따라서 매니코어 시스템에서 Cilk 프로그램의 성능 예측을 위한 확장성 모델을 제안하였다. 제안한 모델에서는 기존 확장성 모델과 새로이 제안되는 NoC 지연시간 모델을 활용하여 다수의 코어에 대한 프로그램의 계산 파트의 분배와 코어 별 NoC 에서의 가변적인 통신 딜레이를 나누어 계산하고자 하였다. 특히, NoC 지연시간 모델에서는 플릿의 종류 별 증가량 예측, 메모리 영역 별 접근 비율 등을 반영하여 보다 정확한 성능 예측을 수행하고자 하였다.

감사의 글

이 연구를 위해 연구장비를 지원하고 공간을 제공한 서울대학교 컴퓨터연구소에 감사 드립니다. 이 논문은 2012 년도 정부(교육과학기술부)의 재원으로 한국연구재단-차세대정보컴퓨팅기술개발사업의 지원을 받아 수행되었습니다. (No. 2011-0020514).

참고문헌

[1] M. Frigo, C. E. Leiserson, and K. H. Randall, "The implementation of the Cilk-5 multithreaded language," In Proceedings of the ACM SIGPLAN, 1998.

[2] J. Reinders, "Intel threading building blocks," O'Reilly & Associates, Inc., 2007.

[3] K. Ebcioglu, V. Saraswat, and V. Sarkar, "X10: Programming for hierarchical parallelism and non-uniform data access," International Workshop on Language Runtimes, 2004.

[4] Y. He, C. E. Leiserson, and W. M. Leiserson, "The cilkview scalability analyzer," ACM Symposium on Parallel Algorithms and Architectures, 2010.

[5] J. H. Ahn, "McSim: a manycore simulation infrastructure," At http://cal.snu.ac.kr/mediawiki/index.php/McSim

[6] V. J. Reddi, A. Settle, D. A. Connors, and R. S. Cohn, "Pin: A Binary Instrumentation Tool for Computer Architecture Research and Education," Workshop on Computer Architecture Education, 2004.

[6] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 Programs: Characterization and Methodological Considerations," In Proceedings of International Symposium on Computer Architecture, 1995.