ESTEREL을 이용한 Cache Coherency Protocol의 정형 설계 및 검증

김민숙° 최진영

고려대학교 컴퓨터학과

{mskim, choi} @formal.korea.ac.kr

Formal Design and Verification of Cache Coherency Protocol by ESTEREL

Min-Suk Kim° , Jin-Young Choi

Dept. of Computer Science and Engineering, Korea University

요 약

캐쉬 일관성 유지 프로토콜은 공유 메모리 다중 프로세서 시스템의 정확하고 효율적인 작동에 중요하다. 시스템이 점점 복잡해짐에 따라 시뮬레이션 방법만으로는 프로토콜의 정확성을 확인하기는 어렵다. 본 논문에서는 CC-NUMA용 디렉토리 기반 캐쉬 일관성 프로토콜인 RACE 프로토콜을 정형기법 도구인 ESTEREL을 이용하여 프로토콜이 안정적으로 동작함을 검증하였다.

1. 서 론

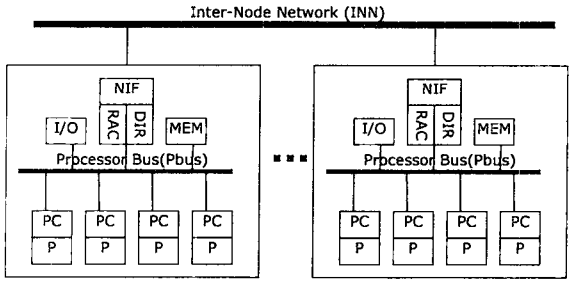
컴퓨터를 이용하는 응용 분야들 중에는 현존하는 초고속 컴퓨터의 성능으로도 처리 시간이 매우 오래 걸리는 것들이 많이 있다. 컴퓨터의 속도를 결정하는 첫 번째 요소인 프로세서의 속도가 계속 향상되고는 있지만 필요한 정도의 시스템 성능을 얻기에는 여전히 부족하다. 따라서 최근 대부분의 컴퓨터 설계에서는 성능 향상을 위한 방법으로서 병렬처리 기술이 널리 사용되고 있다. 공유-기억장치 구조를 가진 병렬 처리 컴퓨터에서는 여러 개의 프로세서들이 동시에 기억장치를 액세스하는 경우가 빈번히 발생하기 때문에 기억장치 충돌에 의한 지연이 발생한다. 이 때 프로세서들간의 여러 개 데이터 복사본 사이의 일관성을 유지하기 위해서, 캐쉬, 메모리 등의 통신 개체들을 조정하는 일종의 규칙을 캐쉬 일관성 프로토콜이라 한다. 그러나 시스템이 점점 복잡해짐과 비례하여 캐쉬 일관성 유지 프로토콜 또한 복잡해지기 때문에 무작위적 테스트나 시뮬레이션은 프로토콜의 정확성을 확인하기에 충분하지 못하므로, 효율적이고 믿을 만한 검증 방법이 필요하다. 정형 기법[1]을 이용하면 자연어가 내포할 수 있는 애매 모호함이나 불확실성을 최소한으로 줄여 시스템을 구현하기 이전에 그 기능의 정확성을 검사할 수 있어 개발비용과 시간을 줄일 수 있다. 하지만 정형기법을 도입한 설계 및 검증은 수학적 기호나 전문 용어로 기술된 사용자의 요구 명세를 검증한 경우로, 전문가의 설명이나 지식 없이는 이해하기 쉽지 않아 그 효용도가 낮은 편이었다.

본 논문에서 이에 대해 일반 사용자들이 이미 친숙한 범용 언어와 유사한 ESTEREL[2]을 이용하여 ETRI에서 개발한 캐쉬 일관성 프로토콜인 RACE프로토콜을 검증한 사례를 제시한다. 2장에서는 RACE프로토콜에 대해, 3장에서는 명세 및 검증 도구로 사용한 ESTEREL toolset에 대해, 4장에서는 ESTEREL을 이용한 RACE 프로토콜의 명세 및 검사, 검증에 대해 논한다. 끝으로 5장에서는 결론을 맺는다.

2. RACE 프로토콜

2.1 개요

CC-NUMA(Non-Uniformed Memory Access)의 일종인 PDLSystem(Physically-Distributed but Logically-Shared memory system)은 [그림1]과 같이 inter-node 네크워크에 연결되어 있는 많은 처리 노드들로 구성되어 있다. 처리 노드는 개인 캐쉬를 갖는 프로세서와 전체 공유 메모리의 일부분인 메모리 모듈, I/O, 외부 접근 캐쉬와 디렉토리를 갖는 네트워크 인터페이스를 갖는다.



[그림1] PDLSystem: DIR(Directory), RAC(Remote Access Cache), NIF(Network Interface), P(Processor), PC(Processor Cache)

노드 안의 프로세서 캐쉬들은 snoopy방식의 일종인 MESI프로토콜에 의해 일관성이 유지되며, 노드 간의 이관성은 Directory 기반의 RACE(Remote Access Cache coherency Enforcement Protocol) 프로토콜에 의해 유지된다. 네트워크 관점에서 RACE

protocol을 설명하면 크게 Local accesses와 Remote Accesses의 두 가지로 나눌 수 있다. 이 중 검증의 대상이 되는 Local accesses에 대해 설명하면 다음과 같다.

2.2 Local accesses

2.2.1 coherent remote request

- 캐쉬 라인을 읽기 전용으로 요구하는 경우
- 요구한 프로세서는 데이터를 수정하지 않음
- RAC가 Shared상태인 경우 RAC이 요청 데이터를 공급
- RAC가 Modified상태인 경우 RAC이 요청 데이터를 공급

2.2.2 exclusive remote request

- 캐쉬 라인을 쓰기 가능으로 요구하는 경우
- 요구한 프로세서는 데이터를 수정 할 수 있음
- RAC가 Shared상태인 경우 무효화 요청 메시지를 홈 노드에 보내고 홈 노드는 그 요청을 받자 마자 모든, 공유 노드에 무효화 요청을 보냄
- RAC가 Invalidate상태인 경우 쓰기 요청 메시지를 홈 노드에 보냄
- 홈 노드가 uncached상태이면 디렉토리는 프로세서 캐쉬로부터 독점적 복사본을 받아 요청 노드에게 보냄
- 홈 노드가 Modified상태이면 디렉토리는 요청을 점유 노드에 전달하고 점유 노드의 RAC가 그 요청을 받자마자 프로세서 캐쉬로부터 가장 최근의 수정 데이터를 가져와 요청 노드에게 보냄.

2.2.3 Invalidate remote request

- 다른 모든 캐쉬의 복사본을 무효화함
- RAC가 Shared상태인 경우, 무효화 요청 메시지를 홈 노드에 보내고, 홈 노드는 무효화 요청을 모든 공유 노드에 보내고 홈 노드의 프로세서 캐쉬를 무효화 시킴

remote accesses에는 위의 requests뿐 아니라 write-back remote request도 포함된다.

3. 정형 기법 도구

3.1 ESTEREL

ESTEREL은 프랑스의 Sophia-INRIA에서 연구 개발되고 있는 정형 명세 및 검증 toolset이다. ESTEREL 언어는 하드웨어나 프로토콜 등을 명세하기 위해 개발된 언어로 Reactive 시스템의 동기적 특성을 표현할 수 있다. 이러한 동기적 언어는 완벽한 동시성 모델에 근거하며, 이러한 모델 내에서는 동시적 프로세스가 0시간 내에 계산과 정보의 교환이 일어나며, 입력 set에 대한 모든 반응은 시간을 소모하지 않는 instantaneous한 것으로 간주된다. 또한 프로세서들간의 통신은 Broadcasting 기법에 의해 실현된다. ESTEREL은 high-level 기술언어와 완전 자동 검증기로 구성된다. 기술 언어는 assignment, if-then-else, present-case, procedure call등과 같이 Pascal이나 C와 같은 프로그래밍 언어에서 볼 수 있는 많은 문장들을 제공한다. 이러한 점에서 high-level언어에 익숙한 사용자들에게 많은 장점을 제공한다. ESTEREL언어는 변환된 시뮬레이션용 C코드를 XES에 입력[4]하여 그래픽적으로 시뮬레이션을 해보거나, BLIF(Berkely Logical Interchange Format)로 변환된 코드를 XEV에 입력하여 검증해 볼 수 있다. 이러한 기법을 이용하면 설계 단계에서 오류를 찾을 수 있기 때문에 오류가 없는 설계를 통해 구현자가 보다 완벽한 구현을 할 수 있도록 할 수 있다.

3.1 XEVE(an ESTEREL Verification Environment)

XEVE[5]라는 도구는 implicit 하게 정의된 유한 상태 기계를 대상으로 모델 체킹 기법을 이용하여 시스템을 검증할 수 있다. FSM은 BLIF(Berkeley Logical Interchange Format) 형태로 표현된다. XEVE 는 두 가지 검증 기능을 제공한다. 첫 번째는 등가 관계를 이용해 최소화된 유한 상태 기계를 이용하는 것이고 두 번째는 위배 신호의 발생 가능 여부를 확인하는 것이다. 만약 시그널이 발생 가능한 경우라면 XES를 통해 그 경우를 확인해 볼 수 있다.

4. RACE 프로토콜의 정형 명세 및 검증

4.1 RACE 프로토콜의 정형 명세

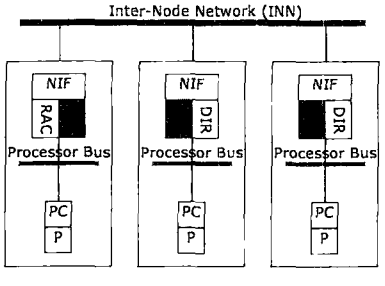
4.1.1 가정

ESTEREL을 이용한 검증은 다음과 같은 가정을 따른다.

- 검증의 범위를 하나의 cache line에 대한 것으로 제한한다. 서로 다른 cache line은 서로의 상태전이에 영향을 미치지 않기 때문에 하나의 cache line에 대한 검증으로 이 모델에 대한 검증은 충분하다고 할 수 있다.
- DIR이나 RAC이 자신의 노드에 있는 메모리, 또는 프로세서 캐쉬와 주고 받는 요청, 응답은 pending상태에 1 tick 머무르는 동안 처리된다고 가정한다..

4.1.2 추상화 모델

[그림 2]는 4.1.1에서 설명한 가정을 기반으로 해서, [그림1]의 PDLSystem을 추상화 한 것이다. 하나의 cache line을 검증 대상으로 하여, 그림에서 왼쪽의 노드를 홈 노드라고 하고, 나머지 두 개의 노드는 원격 노드 1,2가 된다. 따라서 홈 노드의 DIR과 두 원격 노드의 RAC, 그리고 각 RAC에서 오는 요청들과 DIR의 응답들을 처리하는 Inter-Node Network를 구현한다.



[그림 2] ESTEREL에서 사용된 추상화 모델

4.1.3 명세

이를 ESTEREL 언어를 사용하여 명세한 결과는 [그림 3]과 같다. 2개의 RAC에서 입력되는 신호들이 입력되면 자신의 상태를 변경하고 INN에게 그 정보들을 넘겨준다. 그러면 홈 노드와 다른 원격 노드들은 신호를 받아 0시간 내에 자신의 상태를 변동시켜 전체 노드들간의 일관성을 유지시킨다.

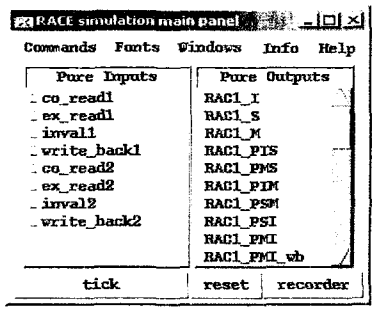
```
module RACE:
input co_read1, ex_read1,  inval1, write_back1;
output RAC1_I, RAC1_S, RAC1_M;
output RAC1_PIS, RAC1_PMS, RAC1_PIM, RAC1_PSM, RAC1_PSI,
       RAC1_PMI, RAC1_PMI_wb;
output DIR_U,DIR_S, DIR_M;
output DIR_PUS, DIR_PSS, DIR_PMS, DIR_PUM,
       DIR_PSM_inv, DIR_PSM_erd, DIR_PMM,
       DIR_PSU, DIR_PMU;
input co_read2, ex_read2,  inval2, write_back2;
output RAC2_I, RAC2_S, RAC2_M;
output RAC2_PIS, RAC2_PMS, RAC2_PIM, RAC2_PSM,
       RAC2_PSI, RAC2_PMI, RAC2_PMI_wb;
   run RAC1/ RAC
||
   run RAC2/ RAC
||
   run DIR
||
   run INN
end module
```

[그림3] ESTEREL로 명세한 RACE프로토콜의 일부

4.2 RACE 프로토콜의 정형적 검사 및 검증

4.2.1 검사

ESTEREL로 명세 된 RACE프로토콜은 XES를 통해 [그림 4]와 같이 각각의 경우를 검사해 볼 수 있다.



[그림4] XES를 이용한 simulation 화면

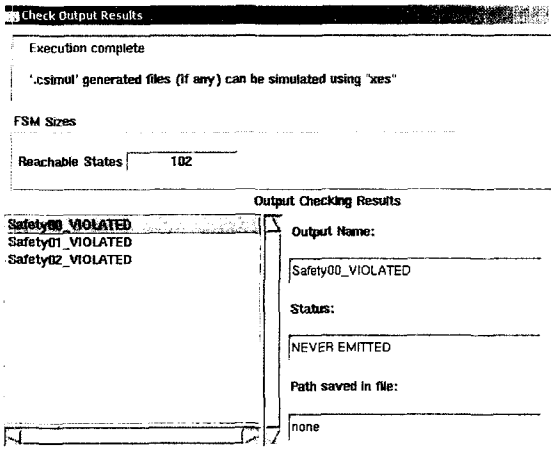
4.2.2 검증한 Property

본 논문에서 RACE프로토콜이 만족해야 할 특성은 다음과 같다.

- Safety00: 두 remote 노드가 동시에 수정 노드 일수 없다.
  -> Always not (RAC1_M and RAC2_M)
- Safety01 : 한 remote 노드가 수정 노드이면, 다른 remote 노드는 공유 노드가 될 수 없다.
  -> Always not ((RAC1_M and RAC2_S) or (RAC1_S and RAC2_M))
- Safety02::홈 노드가 유효노드이면, 다른 remote노드는 공유나 수정 노드일 수 없다.
  -> Always not ((RAC1_S and DIR_U) or (RAC1_M and DIR_U) or(RAC2_S and DIR_U) or (RAC2_M and DIR_U))

4.2.3 검증한 결과

위의 Property를 XEVE를 통해 검증하면 [그림 5]와 같은 결과를 얻는다. 만약 Property를 만족한다면, Safety00_VIOLATED 신호는 절대 발생하지 않을 것이며, Property를 만족하지 못한다면, XEVE는 역 추적 가능한 로그 파일을. 만들어 준다. 따라서, RACE 프로토콜은 3개 이하의 노드를 갖는 PDLSystem에서 Safety 특성을 만족한다는 것을 검증하였다.



[그림5] XEVE 이용한 검증 결과

5. 결론 및 향후 연구과제

CC-NUMA용 디렉토리 기반 캐시 프로토콜인 RACE Protocol를 정형 검증 도구인 ESTEREL tool set을 이용해서 명세 및 검사, 검증해 보았다. XEVE를 통한 검증을 통해 3개의 노드로 구성된 PDLSystem에서 RACE프로토콜이 Safety 특성을 만족함을 보였다. 이러한 시도는 기존의 모델 체킹으로 검증한 결과[6]와 일치하고 사용자에게 좀 더 친숙한 명세 언어를 사용하여 좀 더 쉽게 설계에 적용시킬 수 있다. 또한 XES를 통한 시뮬레이션을 통해 좀 더 명확한 결과를 찾아 낼 수 있었다.

향후 명세 된 ESTEREL코드를 VHDL이나 verilog와 같은 하드웨어 명세 언어로 변환, 합성 과정을 거쳐 현재 사용되고 있는 코드와 비교해 보고자 한다.

참고 문헌

[1] Andrew Harry, *Formal Methods-FACT FILE, VDM and Z*, Wiley, 1996

[2] G.Berry, *The Esterel v5 Language Primer Version v5_91*, INRIA,June 5, 2000

[3] Ki, Ando., *et al.*, *RACE Protocol, Ver 1.1*, Computer System Dept. Computer Software Tech. Lab. ETRI TM-3100-1999-012, V1.1 Draft, Aug. 1999.

[4] G.Berry and the Esterel Team, *The Esterel v5_91 System Manual*, INRIA, June 5, 2000

[5] Amar Bouali, *XEVE : an Esterel Verification Environment(Version v1_3)*, INRIA, December 1997

[6] 남원흥, "SMV를 이용한 RACE 프로토콜과 CCA보드의 정형 검증 및 테스팅", 2001