RAID 시스템의 모델링 및 시뮬레이션

이찬수* · 성영락** · 오하령***

Modeling and Simulation of a RAID System

Lee, Chan-Su, Seong, Yeong-Rak and Oh, Ha-Ryoung

Key Words: RAID, DEVS 형식론, 시뮬레이션, 데이터저장시스템

Abstract

In this paper, a RAID system is modeled and simulated by using the DEVS formalism. The RAID system interacts with a host system by using the high-speed Fibre channel protocol and stores input data in an array of IDE disks. The DEVS formalism specifies discrete event systems in a hierarchical, modular manner. The RAID system model is composed of three units: primary-PCI unit, secondary-PCI unit and CPU unit. The primary-PCI unit interfaces with the host system and caches I/O data. The secondary-PCI unit includes disks. The CPU unit controls overall system. The control algorithm of CPU and PCI transactions are analyzed and modeled. From an analysis of simulation events, we can conclude that the proposed model satisfies given requirements.

1. 서 론

여러 개의 디스크에 동시에 접근함으로써 여러 개의 디스크가 하나의 빠르고 큰 디스크로 보이게 하는 RAID(Redundant Array of Inexpensive Disks) 시스템[1]에 대한 많은 연구가 진행되고 있다. 기존의 연구에서는 스케쥴링 알고리즘 및 매핑 알고리즘[2,3]과 다양한 부하에서 디스크 구성에 따른 성능 분석[4]에 대해서 연구가 진행되었다.

큰 대역폭이 요구되거나 실시간 운영이 요구되는 RAID 시스템은 평면 구조로는 필요한 대역폭을 만족시킬 수 없어 계층적 구조의 형태를 가지는 경우가 많다. 이런 시스템은 성능에 영향을 미치는 여러 가지 파라미터들이 서로 연관되어 최적값을 선택하기 어려울 뿐 아니라 그 성능을 예측하기가 쉽지 않아서 제품 개발 시간과 더불어 최적 시스템을 구현하기가 어렵다.

그러므로 RAID 시스템의 성능을 빠르고 쉽게 예측하기 위하여 형식론적인 시뮬레이션을 필요로 한다. 본 논문에서는 DEVS 형식론을 이용하여 제안된 시스템을 기술하고, 이산사건 시뮬레이션 언어중의 하나인 DEVSim++[5]을 이용하여 시뮬레이션 하였다.

DEVS 형식론은 이산 사건 시스템을 기술하는 수학적 형식론으로서 계층적이고 모듈화한 방법으로 시스템을 기술한다[6]. 이때 각각의 구성요소들은 원소형 모델로 표현되며 계층적인 구성은 결합형 모델로 나타낸다. 원소형 모델은 외부 사건 또는 내부 시간 초과에 의해 상태가 천이 하는 동적 모델에 해당하며, 출력은 내부 시간 초과에 의해

* 국민대학교 전자공학과 대학원 박사과정 수료
** 국민대학교 전자공학부 조교수
*** 국민대학교 전자공학부 교수

상태가 천이 될 때에 발생된다. 이러한 출력은 다른 모델의 외부 사건을 발생시켜 다른 모델의 상태 천이를 발생시킨다. 결합형 모델은 그 인터페이스는 원소형 모델과 같지만 구성은 원소형 모델 또는 다른 결합형 모델의 집합으로 되어 있다. 각 모델들은 입력과 출력으로 연결이 되며, 유한한 상태를 가지고, 스스로 발생시키는 사건과 외부에서 발생하는 수동적인 사건들로 상태가 천이 된다.

본 논문에서는 IDE 타입의 디스크를 이용하며, 외부 호스트와는 FC(Fibre Channel) 인터페이스를 통해 고속으로 데이터를 입출력할 수 있는 RAID 시스템의 동작특성을 추상화하고 개념화하여 DEVS 형식론으로 기술한다.

2. RAID 시스템

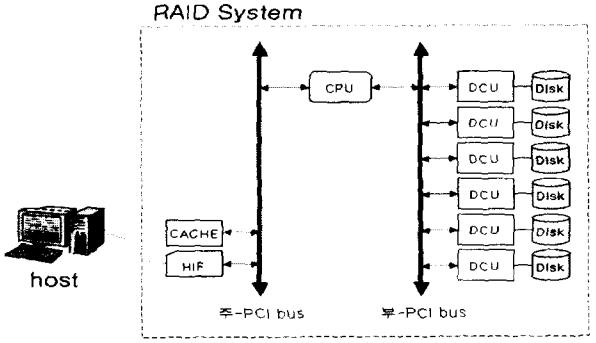

그림 1 RAID 시스템 구조

본 논문에서 모델링 대상으로 삼는 RAID 시스템은 그림 1과 같은 구조를 가진다. 시스템은 크게 CPU부, 주-PCI부, 부-PCI부로 구성된다. CPU부는 데이터를 입출력 할 수 있도록 전체 시스템을 제어하는 역할을 담당한다. 주-PCI부는 FC와 메모리, 그리고 PCI 버스로 구성되며 외부 호스트와의 입출력 및 입출력 데이터를 메모리에 캐싱 하는 기능을 담당한다. 부-PCI부는 실제 디스크와 PCI 버스로 구성되어 실제 디스크 입출력을 담당한다. 저장매체로는 비교적 저렴한 가격으로 널리 사용되고 있는 IDE 디스크를 이용한다. 각각 디스크는 디스크 컨트롤러(DCU)에 연결되어 두 디스크를 병렬로 운용하여 높은 디스크 입출력 성능을 갖도록 한다. 각각의 DCU는 PCI 버스를 통해 CPU와 연결하도록 한다.

시스템은 외부 호스트와는 FC 인터페이스를 통해 고속으로 데이터를 입출력할 수 있도록 한다. FC[7]는 기존의 SCSI 프로토콜로 호스트와 입출력하므로 별다른 호스트 드라이버의 개발 없이도 바로 기존 시스템에 사용할 수 있고, 전송거리가 기존 인터페이스와는 달리 최대 10km 까지 가능하기에 추후에 SAN(Storage Area Network), NAS(Network Attached Storage) 등 네트워크 저장 장치[8]의 지원을 가능하게 하는 발판을 마련한다. 그리고 제안하는 시스템에는 2Gbyte의 메모리를 장착하여 다량의 데이터가 잦은 입출력을 가질 때 발생하는 디스크 버퍼 캐쉬, 파일 시스템 캐쉬의 미스율(miss ratio)을 개선할 수 있도록 한다. 운영방식으로는 RAID0~RAID6 까지 다양한 방법으로 구성할 수 있으나 본 논문에서는 패리티를 가지며 디스크 스트라이핑을 하는 RAID5 방식으로 구현한다.

3. 시스템 모델링

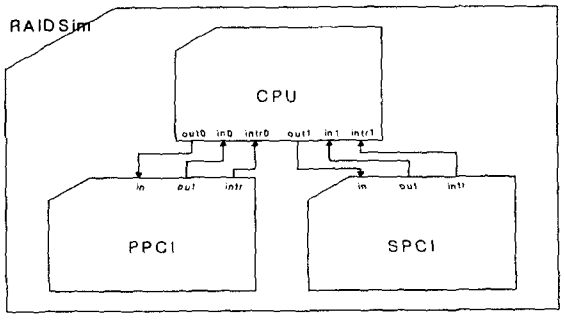

그림 2 RAID 시스템 모델

본 절에서는 2절에서 제시한 RAID 시스템을 모델링 한다. 그림 2는 그림 1의 시스템을 모델링 한

것이다. 그림 2의 구조와 같이 크게 CPU, PPCI, SPCI로 구성된다. 각각의 상세한 모델은 다음과 같다.

3.1 CPU부

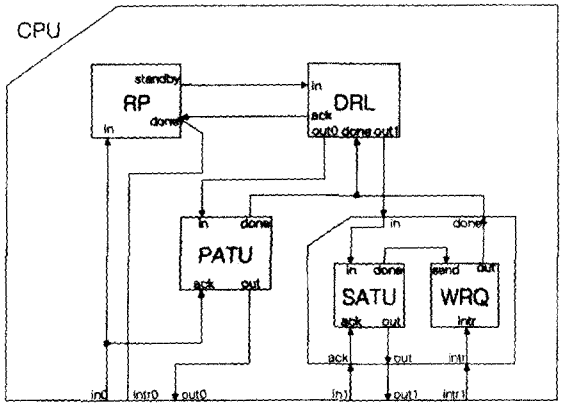

그림 3 CPU 모델 구성도

CPU부는 그 구조와 제어 알고리즘을 개념화하고 추상화하여 그림 3과 같이 모델링 한다. CPU부는 i)CPU내의 SCSI 요청 메시지들을 관리 RP(Request Pool), ii)각 장치별 요청 메시지 리스트를 관리하는 DRL(Device Request List), iii)CPU부와 PPCI와 인터페이스를 담당하는 PATU(Primary ATU), iv)CPU부와 SPCI와 인터페이스를 담당하는 SATU(Secondary ATU), 그리고 v)SATU에서 SPCI로 보낸 메시지에 대해 타겟으로부터 응답 메시지를 받기 전까지 그 DCU를 조작할 수 없기에 각 DCU별로 지연된 IDE 요청 메시지 리스트 관리를 담당하는 WRQ(Wait Response Queue)로 구성된다.

동작 알고리즘을 살펴보면, i)FC로부터 입력받은 SCSI 요청 메시지가 CPU에 들어오면 그것에 대한 IDE 요청 메시지, 메모리 요청 메시지, SCSI 응답 메시지 등 CPU에서 발생시킬 모든 PCI 요청 메시지들을 만들어서 RP에 넣는다. ii)DRL에서는 해당 디바이스가 IDLE 상태이면 첫 번째 요청 메시지를 CPU의 PATU와 SATU의 요청-큐로 보낸다. iii)PATU와 SATU에서는 PCI 버스가 IDLE 상태가 되면 요청-큐에서 요청 메시지들을 꺼내어 해당 타겟으로 보낸다. iv)이 때 PPCI, SPCI로 보내어진 요청 메시지는 각각 DRL과 WRQ로 보내어진다. v)DCU로부터 IDE 응답 메시지가 도착하면 WRQ에서 해당 요청 메시지를 찾아 DRL로 보낸다. vi)DRL에서는 이 응답 메시지를 받으면 해당하는 요청 메시지를 제거한다. vii)제거된 요청 메시지는 RP로 보내어진다. viii)RP에서는 해당 요청 메시지가 삭제됨으로써 처리를 마치게 된다.

3.2 주-PCI부

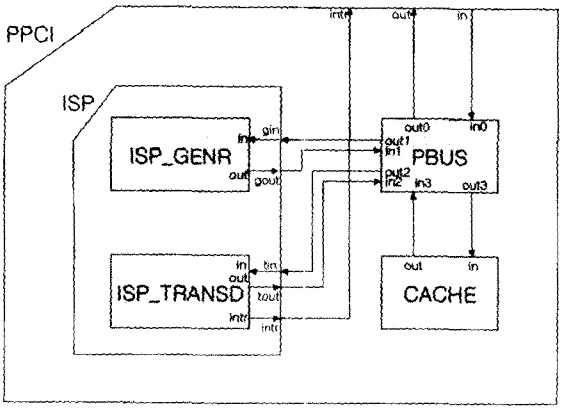

그림 4 주-PCI 모델 구성도

주-PCI부는 그림 4와 같이 ISP, CACHE, PCI 버스(PBUS)로 구성된다. ISP는 FC 프로토콜로 외부 호스트로부터 데이터를 입력받는 역할을 추상화하여, 임의의 간격으로 SCSI 요청 메시지를 발생하여 보내고, 처리되어진 응답 메시지를 받아 요청한 명령이 적절히 처리되었는지 확인할 수 있도록 각각 ISP_GENR와 ISP_TRANSD로 나누어 구현한다. PBUS는 ISP와 CACHE, CACHE와 CPU 간의 데이터 전송을 맡으며, 하나의 명령으로 송수신되는 데이터는 그 전송이 한번의 아비트레이션으로 완료될 수 있도록 다음 타겟으로 스케쥴링 되지 않게 PCI 아비터를 설계하여 사용한다고 가정한다. 디스크 읽기 동작 패러다임은, 디스크로부

터 읽혀진 데이터는 CPU에 의해 CACHE에 쓰여지고, 이것을 ISP가 읽어가도록 한다. 디스크 쓰기 동작은 반대의 과정을 거친다.

3.3 부-PCI부

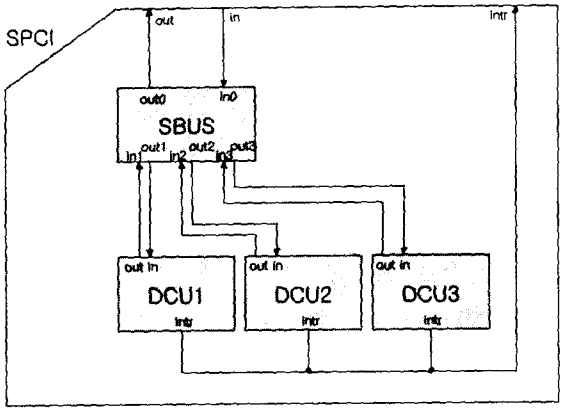

그림 5 부-PCI 모델 구성도

그림 1의 RAID 시스템에서 DCU의 개수는 시스템의 구성에 따라 1~6개까지 가능하다. 본 논문에서는 설명을 간단하게 하기 위해 3개의 DCU가 연결된 구성에 대해서 모델링 한다. 추후에 보다 실제적인 시스템의 시뮬레이션을 위해서는 DCU의 갯수를 조절하면 된다. 또한 하드디스크의 동작 특성을 DCU 모델에 포함하여 모델링 한다.

4. PCI 트랜잭션 모델링

전체 시스템은 모두 19개의 모델로 구성되며, 전체 모델에 대한 기술은 너무 방대하므로 본 논문에서는 지면 관계상 그 중에서 시스템 각 부분을 연결하여 각종 메시지를 전달하는 역할을 담당하는 PCI 버스의 DEVS 모델에 대해서만 살펴보겠다.

PCI 버스 모델은 각각의 이니세이터로부터 PCI 명령을 받아들이는 입력포트 in0~in6, 입력된 명령을 해당 타겟으로 전달하는데 사용되는 출력포트 out0~out6을 가지고 있다. 하나의 PCI 버스에는 최대 7개의 디바이스가 연결될 수 있다고 가정한다. 그림 6은 PCI 버스 모델의 상태 천이도를 나타낸다.

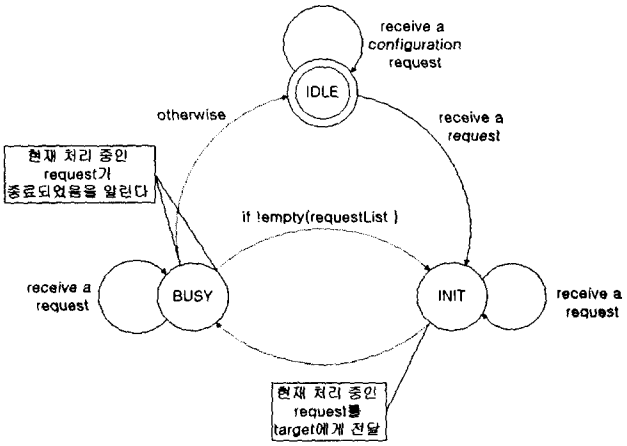

그림 6 PCI 버스 모델의 상태 천이도

시뮬레이션 초기화(t=0) 때에 각 장치들은 자신의 ID를 버스로 보내주어 PCI 버스의 각각의 포트에 연결된 장치들의 ID를 확인한다. 이는 PCI 버스의 컨피규레이션 사이클을 모델링 한 것이다. 이후 평소에는 IDLE 상태이며, 버스 요청이 들어오면 INIT 상태로 천이하며 버스 아비트레이션을 수행한다. 버스 허가시 BUSY 상태로 천이하며 BUSY 상태가 끝나면 지연된 버스 요청을 확인하여 있으면 INIT 상태로, 없으면 IDLE 상태로 천이한다. INIT 상태가 끝날 때(BUSY 상태가 시작될 때) 버스 요청 내용이 타겟에 전달되고, BUSY가 끝날 때 EndOfTransfer를 이니세이터와 타겟에 전달한다. BUSY 상태의 지속 시간은 초기화 때에 이니세이터와 타겟의 버스폭, 데이터 전송량에 의해서 결정된다.

5. 시뮬레이션 결과

그림 7은 캐쉬 히트율 0% 일 때 3개의 DCU로부터 5개 데이터 블록을 읽어들이는 과정을 시뮬레이션 한 결과 중 SPCI 모델들간에 메시지를 주고받은 과정을 나타내었다. 이 때 PCI 버스 사용

은 우선순위에 기초한 아비트레이션이 일어난다고 가정하였다. 그림에서, DCU1에서 읽은 데이터를 CPU로 보내는 동안(t=362~585) DCU2도 읽은 데이터를 CPU로 보내려 한다. 하지만 이미 DCU1이 PCI 버스를 점유하고 있기에 이 처리가 끝난 t=586까지 대기하게 된다. t=844에서는 아비트레이션이 일어나 대기중인 CPU와 DCU3 중 우선순위가 더 높은 CPU로 버스 사용을 허가하는 것을 알 수 있다.

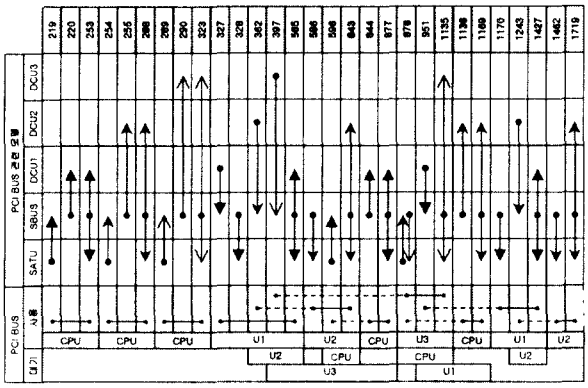

그림 7 시뮬레이션 결과(SPCI부)

6. 결 론

본 논문에서는 IDE 디스크들을 이용하고, FC 인터페이스를 갖는 RAID 시스템을 DEVS 형식론으로 기술하였다. 특히 전체 시스템을 제어하는 CPU와 PCI 트랜잭션의 제어 알고리즘을 추상화하고 개념화하여 기술하였으며, 시뮬레이션 과정에서 발생된 사건들을 분석한 결과 제안한 모델들이 주어진 요구사항을 잘 만족함을 보았다.

본 논문에서 만들어진 모델을 이용하면 RAID의 여러 구성을 손쉽게 모델링 하여 시뮬레이션 할 수 있으며, 각종 파라미터 조정과 시스템 구조 변경으로 인한 성능 측정을 통해 좀 더 효율적인 시스템을 구축하는데 활용될 수 있다. 추후과제로 실제 파일 시스템 로그 데이터를 이용하여 좀 더 실제적인 디스크 입출력 데이터를 이용한 성능 분석과 제안된 시뮬레이터를 바탕으로 효율적인 RAID 시스템의 구조에 대한 연구가 진행중이다.

참고문헌

[1] G. R. Ganger, B. L. Worthington, R. Y. Hou, and Y. N. Patt, "Disk arrays," IEEE Computer, pp. 30-36, March 1994.

[2] Shenze Chen and Don Towsley, "A Performance Evaluation of RAID architecture," IEEE Trans. Computers, Vol. 45, No. 10, 1996.

[3] Arif Merchant and Philip S. Yu, "Analytic Modeling of Clustered RAID with Mapping Based in Nearly Random Permutation," IEEE Trans. Computers, VOl. 45, No. 3. 1996.

[4] A. L. Narasimha Reddy and Prithviraj Banerjee," An Evaluation of Multiple-Disk I/O Systems," IEEE Trans. Computers, Vol 38, No. 12, 1989.

[5] 안명수, 박성봉, 김탁곤, "DEVSim++: 의미론에 기반한 이산사건 시스템의 객체지향 모델링 및 시뮬레이션 환경," 한국정보과학회논문지, 제21권, 제9호, pp. 1652-1664, 1994.

[6] Bernard P. Zeigler, "Object-Oriented Simulation with Hierarchical, Modular Models," Academic Press, 1990.

[7] Fibre Channel Standards, http://t11.org

[8] William Stallings, Computer Organization and Architecture, 5th, ED., Prentice Hall, 2000.