RAID 시스템의 성능 평가

이찬수* · 성영락** · 오하령***

A Performance Evaluation of a RAID System

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

Abstract

본 논문은 RAID 시스템의 몇 가지 구성에 대해 시뮬레이션을 통해 그 성능을 분석한다. 대용량의 RAID를 구성하기 위해서는 다수의 디스크가 필요한데 반해, 하나의 PCI의 버스에 연결될 수 있는 장치의 개수 제한되어 있어 이 경우 확장 PCI 버스가 필요하다. 본 논문에서는 RAID 시스템의 하드웨어 구조, 특히 각 구성요소를 연결하는 PCI 버스의 부하에 초점을 맞춘다. 버스 트랜잭션을 세 가지로 분류하고, 각각의 경우를 분석하고 평가한다. 이 분석으로부터 RAID 시스템의 두 가지 구조에 대해 성능을 계산하고 시뮬레이션한 결과를 비교한다.

Key Words : RAID, DEVS 형식론, 시뮬레이션, 데이터저장시스템

1. 서론

최근 들어 인터넷의 보급이 확산되면서 웹디스크의 형태로 디스크 저장 공간을 사용자에게 제공하는 웹사이트들이 늘어가고 있다. 뿐만 아니라 산업 전 부분이 인터넷 환경으로 전환되면서 개인 데이터는 물론 기업의 각종 데이터가 기하급수적으로 증가하고 있다. 이처럼 많은 사용자들의 다양한 데이터 입출력 요구를 효과적으로 처리하기 위해서는 고속, 고신뢰성의 디스크 저장 시스템이 요구된다. SAN(Storage Area Netowork) 시스템은 이러한 요구사항에 맞게 개발된 시스템으로서 현재 많은 웹사이트들이나 기업의 서버들은 SAN을 이용한 시스템을 구축하여 운영중이다[1].

본 논문에서는 SAN의 기본이 되는 RAID 시스템의 성능을 분석한다. RAID는 디스크 어레이(Disk Array)에 부가적인 데이터를 저장하기 위한 디스크들을 추가하여 신뢰성을 강화한 것이다. 그 동안 많은 연구를 통해서 RAID는 수준 0에서 수준 6까지 다양한 방식이 존재한다[2]. 본 논문에서는 RAID 수준 5에 중점을 두고 있다. 이 방식은 현재 가장 널리 사용되는 RAID 방식중의 하나로서 특히 여러 사용자들의 입출력 요구를 처리하는 데에 적합한 방식이다.

본 논문에서는 기존의 연구[3-5]와는 달리 RAID 시스템의 하드웨어 구조에 중점을 둔다. 특히 하드웨어 구성 요소들을 연결하는 버스 시스템에서의 부하를 분석한다. 이것을 통하여 구성요소들의 구성 형태에 따른 전체 시스템의 성능을 예측해 보고, 시뮬레이션을 통해 그 타당성을 검증해 보고자 한다. 본 논문에서는 두 가지의 RAID 시스템 구성 예에 대해서 성능을 비교한다.

본 논문의 구성은 다음과 같다. 2장에서는 본

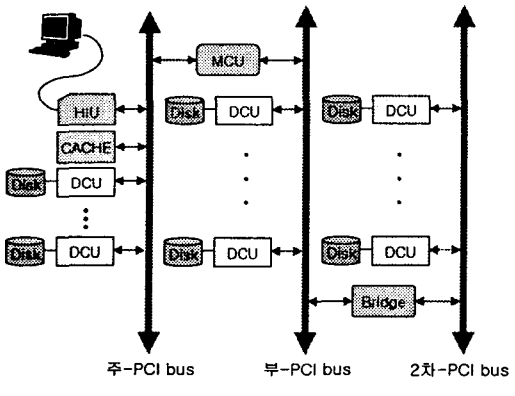
* 국민대학교 전자공학과 대학원 박사과정 수료
** 국민대학교 전자공학부 부교수
*** 국민대학교 전자공학부 교수

논문에서 분석할 RAID 시스템의 하드웨어 모델과 각 구성요소들의 역할을 살펴본다. 3장에서는 대상 시스템에서 주요 구성 요소들간의 최대 데이터 전송 속도를 분석한다. 4장에서는 두 가지 시스템 구성 예에 대해서 3장의 결과를 가지고 성능을 비교한다. 마지막으로 5장은 결론이다.

2. RAID 시스템 모델

본 절에서는 본 논문에서 성능 분석의 대상이 되는 RAID 시스템에 하드웨어 모델을 제시한다.

현재까지 매우 다양한 종류의 RAID 시스템이 개발되었다. 그리고 하드웨어 구조 또한 다양하다. <그림 1>은 본 논문에서 성능 분석의 대상이 되는 RAID 시스템의 모델이다. 시스템은 크게 호스트 시스템과의 통신을 담당하는 HIU(Host Interface Unit), 디스크와 인터페이스하는 DCU(Disk Control Unit), 디스크 데이터에 대한 임시 저장 장소인 캐쉬, 시스템 전체를 제어하고 패리티 연산을 수행하는 MCU, 그리고 이들 구성요소들을 연결하는 3개의 PCI 버스로 구성된다.



<그림 1> RAID 시스템 모델

HIU는 Fibre Channel과 PCI 버스의 두 가지 인터페이스를 이용하여 고속으로 호스트 시스템과 RAID 시스템을 연결하는 역할을 한다[1][6]. DCU는 MCU가 PCI 버스를 통하여 디스크 엑세스 명령을 내리면 디스크를 제어하여 디스크와 MCU 사이의 데이터 전송을 중개하는 역할을 한다. 이때 디스크와의 연결은 SCSI, IDE 등의 일반적인 방식이 이용된다.

캐쉬는 RAID 시스템의 성능에 가장 큰 영향을 주는 부분 중의 하나이다. 캐쉬의 역할은 운영체제의 버퍼캐쉬의 복사본들을 저장하여, 디스크 속도와 호스트 인터페이스 속도의 차이를 완충하는 역할을 한다.

PCI 버스는 시스템의 구성 요소들을 연결하는 역할을 한다[7]. PCI 버스의 제약사항으로는 하나의 물리적인 버스에 연결되는 장치들의 수에 대한 제한이 있다. 그래서 연결되어야 할 장치들의 수가 많을 때에는 PCI-to-PCI 브리지를 이용하여 버스를 확장한다. 브리지 양쪽의 버스는 서로 독립적으로 작동된다.

MCU는 전체 시스템을 제어하는 역할을 한다. 또 RAID 수준 5를 위한 패리티를 계산한다.

MCU는 두 개의 PCI 인터페이스를 가지는 것으로 간주하였다.

<그림 1>에서는 구성요소들의 연결 형태만 나타낸 것이고 구체적인 내용들이 포함되어 있지 않다. 그중 하나는 각 구성요소들과 PCI 버스와 인터페이스이고 또 하나는 DCU의 개수이다. 본 논문에서는 HIU, 캐쉬, MCU, 브리지는 64비트 인터페이스를 가지며, DCU는 32비트 인터페이스를 가지는 것으로 가정하였다. 그러므로 DCU와 관련된 데이터 전송은 32비트로 전송되고, 나머지는 64비트로 전송된다. 또 DCU의 개수는 총 10개로 가정하였다. 각각의 버스에 연결되는 DCU의 개수는 4장에서 구체적인 성능 분석시에 제시하겠다.

<그림 2>는 호스트 시스템에서 <그림 1>의 시스템으로 읽기 및 쓰기 명령이 전달될 경우에 각 구성요소들간의 제어 및 데이터의 흐름을 알고리즘으로 나타낸 것이다. 알고리즘 앞에 unit는 알고리즘을 수행하는 시스템 구성 요소를 나타낸 것이다.

```
unit algorithm
(1) HIU  receive a read request from host:
(2) HIU  send the request to MCU:
(3) MCU  if ( the request is read )
(4) MCU     if ( the data is already stored in the cache )
(5) MCU        send the cache address to HIU:
(6) HIU        transfer data from cache:
(7)         else
(8) MCU        send disk read requests to DCUs:
(9) DCU        transfer data to MCU:
(10) MCU       transfer data to cache:
(11) MCU       send the cache address to HIU:
(12) HIU       data transfer from cache:
(13)        endif
(14)     endif
```

<그림 2> 읽기동작에 대한 시스템제어 및 데이터흐름

3. 데이터 전송시간 분석

본 절에서는 RAID 시스템의 각 부분들간의 데이터 전송을 분석한다. <그림 2>를 바탕으로 PCI 버스를 통한 각 부분들의 데이터의 흐름을 살펴보면 다음과 같이 분류할 수 있다.

i) HIU와 MCU 사이
ii) HIU와 캐쉬 사이
iii) 캐쉬와 MCU 사이
iv) MCU와 DCU 사이
v) MCU와 브리지 사이
vi) 브리지와 DCU 사이

i)은 HIU와 MCU 사이에 SCSI 작업 요청과 응답을 위한 데이터의 흐름으로서 크기가 작다. 그것에 비하여 나머지들은 디스크 입출력 데이터들의 흐름으로서 크기가 크다. 본 논문에서는 ii)와 iii)을 묶어서 캐쉬를 통한 MCU와 HIU 사이의 데이터 전송, iv)를 브리지를 통하지 않은 MCU와 DCU 사이의 데이터 전송, v)와 vi)을 묶어서 브리지를 통한 MCU와 DCU 사이의 데이터 전송으로 구분하여 분석하였다.

3.1 캐쉬를 통한 MCU와 HIU 사이의 데이터 전송

MCU와 HIU 사이에서의 데이터의 전송은 반드시 캐쉬를 통하여 이루어져야 한다. 만약 데이터 읽기의 경우에는 MCU가 캐쉬에 데이터를 전송한 다음에 HIU가 캐쉬 데이터를 읽어야 하며, 데이터 쓰기의 경우에는 HIU가 캐쉬에 데이터를 쓰면 MCU가 그 데이터를 읽어서 처리한 후에 DCU로 전달하여야 한다. 이때 전달되는 데이터의 크기가 a 바이트라고 하면 MCU와 캐쉬간에 데이터 전송 시간 $TMCU$와 HIU와 캐쉬간의 데이터 전송시간 $THIU$는 다음과 같다.

$$TMCU = a/8$$
$$THIU = a/8$$

여기서 8로 나뉘는 것은 64비트씩 데이터가 전달되기 때문이다. 또한 a는 디스크 블록 크기에 비해서 매우 큰 값으로 가정한다.

3.2 브리지를 통하지 않은 MCU와 DCU 사이의 데이터 전송

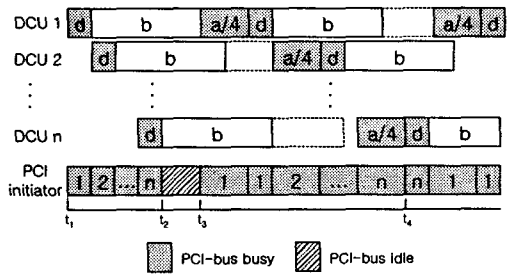
MCU와 DCU 사이의 데이터 전송은 디스크 읽기와 디스크 쓰기 두 가지의 경우가 있다. 그러나 MCU와 HIU 사이의 경우와는 달리 디스크 읽기와 쓰기는 세부적인 동작의 순서만 다를 뿐, 기본적으로는 같은 동작을 취하게 된다.

디스크 읽기를 위해서는 우선 MCU가 DCU의 레지스터들을 설정해야 한다. 그러면 DCU는 설정된 값에 따라 디스크를 엑세스하여 데이터를 읽은 후, PCI 버스를 통하여 MCU로 그 데이터를 DMA 전송한다. 데이터 전송이 끝나면 DCU는 MCU에 인터럽트를 걸어서 데이터의 전송이 끝마쳤음을 알린다. 일반적으로 MCU 내부에서는 많은 디스크 연산들이 대기중일 것이므로, MCU는 다시 DCU의 레지스터들을 초기화하여 다음 디스크 연산을 시작할 것이다. 디스크 쓰기의 경우에는 디스크 엑세스와 PCI를 통한 DMA 전송의 순서만 바뀔 것이다.

<그림 3>은 이 과정들이 여러 DCU에서 병행해서 발생하는 경우를 도식화 한 것이다. 초기에는 DCU에 아무런 작업 요청이 없을 것이고, PCI 버스도 쉬는 상태일 것이다. 그러다가 $t1$에서 MCU가 DCU1의 레지스터를 설정하여 디스크 입출력을 시작할 것이다. 각 DCU의 레지스

터들을 설정하는 데에 걸리는 시간은 d라고 하고, 엑세스되는 DCU의 수가 n개라면, 각 DCU의 첫 번째 레지스터 설정이 끝나는 시각 $t2$는 다음과 같이 정해진다.

$$t2 = t1 + n \times d$$



<그림 3> 브리지를 통하지 않은 CPU와 DIU의 데이터 전송

레지스터가 설정되면, DCU는 설정된 값에 따라 디스크를 엑세스하게 된다. 레지스터 설정이 끝난 직후부터 PCI 버스로 데이터 전송 준비가 끝나는 시점까지의 시간을 b라고 정의한다. 여기에는 DCU가 디스크의 장치 레지스터들을 엑세스하는 시간, 디스크 헤드의 탐색(seek) 및 회전(rotate) 시간이 포함된다. DCU1의 레지스터 설정이 끝나고 b시간이 지나고 나면, DCU1은 PCI 버스로 데이터를 보낼 준비가 된다. PCI 버스로 데이터 전송을 시작하는 시점을 $t3$라고 하면

$$t3 = \begin{cases} t1+nd & \text{if } b \le (n-1)d \\ t1+nd+c & otherwise \end{cases}$$

이다. 여기서 $c(=b-(n-1)d)$는 버스가 휴식상태로 보내는 시간이다. 이때 전송되는 데이터의 양이 a라고 하면 데이터 전송 시간은 $a/4$이다. 이것은 앞서 언급한 대로 DCU의 버스 폭이 32비트이기 때문에 동시에 4바이트씩 전송되기 때문이다. 각각의 데이터 전송이 끝나면 MCU가 다음 작업을 DCU에 요청하기 위하여, 다시 d시간을 소요할 것이다. 그러므로 PCI 버스는 $t3$이후에야 비로소 정상상태가 될 것이다.

정상상태에서 DCU 동작들의 한 주기(그림에서 [$t3$, $t4$] 구간) $TDCU,1$은 다음과 같다.

$$T_{DCU,1}(n) = MAX[b+d+a/4, n(d+a/4)]$$

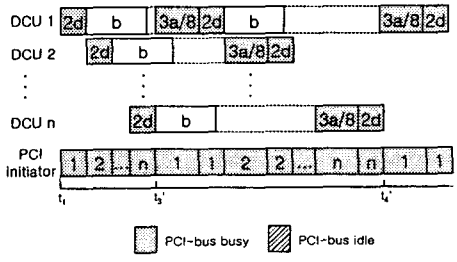
즉, $b<(n-1)(d+a/4)$이면 $TDCU,1$은 디스크 엑세스 시간 b에 무관하게 된다. 이것은 DCU의 레지스터를 설정한 이후부터 DCU가 데이터를 보내기 시작할 때까지의 시간(b) 보다 $n-1$개의 다른 DCU들을 설정하는 데에 걸리는 시간이 더 커지므로, b가 DCU의 DMA 데이터 전송에 전혀 영향을 주지 못하게 되어 DMA 데이터 전송이 완전히 파이프라인 형태로 이루어짐을 의미한다. 일반적으로 $d \ll a/4$ 이므로, DMA 데이터 전송이 완전히 파이프라인 형태로 이루어 질 경우

$$T_{DCU,1}(n) \approx na/4$$

이다.

3.3 브리지를 통한 MCU와 DCU 사이의 데이터 전송

<그림 4>는 브리지를 통한 MCU와 DCU간의 데이터 전송을 도식화 한 것이다. 그림은 디스크 읽기의 경우에 대한 것이지만 3.2절에서 언급한대로 디스크 쓰기의 경우에도 순서만 바뀔 뿐 큰 차이가 없다.



<그림 4> 브리지를 통한 CPU와 DCU 데이터의 전송

<그림 3>의 경우와는 달리 레지스터 설정시

간이 $2d$이다. 이것은 MCU에서 브리지로 설정 데이터를 보내는 시간 d에 브리지에서 DCU로 그 데이터를 보내는 시간 d가 합해진 값이다. 또 a바이트의 데이터를 DCU에서 MCU로 DMA 전송하는 데에 걸리는 시간은 $3a/8$이다. 이것은 브리지가 64비트 데이터 전송능력을 가진 것으로 가정하였기 때문에, DCU에서 브리지까지는 $a/4$ 시간 걸리지만 브리지에서 MCU까지는 $a/8$ 시간이 걸리기 때문이다.

이러한 값들을 가지고 3.2절에서와 같은 방법으로 분석하며, 브리지가 있는 상태에서 정상상태의 DCU 동작들의 한 주가 $TDCU,2$는 다음과 같다.

$$T_{DCU,2}(n) = MAX[b+2d+3a/8,\ n(2d+3a/8)]$$

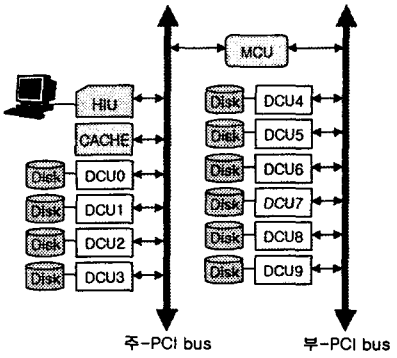
이 경우, b<(n-1)(2d+3a/8)이면 $TDCU,2$는 디스크 엑세스 시간 b에 무관하게 된다. 일반적으로 $2d \ll 3a/8$이므로, DMA 데이터 전송이 완전히 파이프라인 형태로 이루어질 경우

$$T_{DCU,2}(n) \approx 3na/8$$

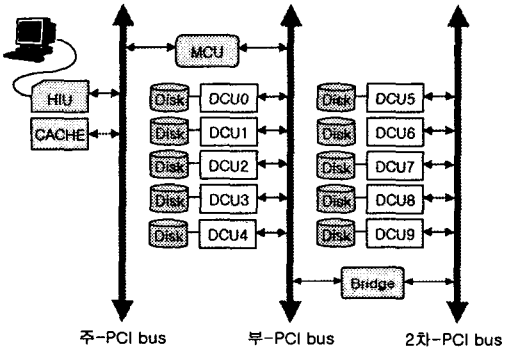
이다.

4. 성능 분석

<그림 1>에서 각각의 PCI 버스에 DCU의 수를 배치하기에 따라서 여러 가지 다양한 시스템 구성이 가능하다. 본 절에서는 3절의 결과를 바탕으로 두 가지 구성의 RAID 시스템들에 대해서 성능을 비교 분석한다. <그림 5>는 성능 분석에 사용된 시스템의 구성이다. 구성 1은 MCU의 주-PCI 버스에는 HIU, 캐쉬, 4개의 DCU가 있고, 부-PCI 버스에는 6개의 DCU가 있는 경우이다. 구성 2는 주-PCI 버스에는 HIU와 캐쉬가 있고, 부-PCI 버스에는 5개의 DCU와 PCI-to-PCI 브리지가 있고, 또 브리지를 거쳐 이차-PCI 버스에 5개의 DCU가 있는 경우이다. 두 구성을 비교해보면 구성 1에서는 버스의 수가 최소화된 것이 특징이며, 구성 2에서는 HIU, 캐쉬, MCU 사이의 버스 대역폭과 작업량의 분산을 고려한 것이 특징이다.



(a) 구성 1



(b) 구성 2

<그림 5> 성능 분석 시스템의 구성

각각의 경우에 대해서, 2개씩의 DCU는 대기 상태인 것으로 가정하였다. 10개의 DCU 모두를 가지고 RAID를 구성하여도 무방하지만 대개의 RAID 시스템의 경우 대기 상태의 디스크들을 두는 것이 추세이다. 구성 1에서는 부-PCI 버스의 2개의 DCU가, 구성 2에서는 부-PCI 버스와 이차-PCI 버스에서 각각 1개씩의 DCU가 대기 상태라고 가정하였다.

각각의 구성에 대해서 분석결과 호스트에서 디스크 쓰기에 대해서는 같은 성능을 보였기에 디스크 읽기 명령이 오는 경우에 대해 분석한 결과는 나타내었다. 그리고 읽기에서 캐쉬히트가 발생하는 경우는 다루지 않았는데, 이것은 양

쪽 구성에서 동등한 조건이 되기 때문이다. 그러나 RAID 시스템이 운영되는 환경에 따라 다르겠지만 대부분의 경우 사용자가 요청하는 작업은 읽기 엑세스이고, 대부분 캐쉬 히트일 것이다.

4.1 시스템 구성 1의 성능 분석

호스트에서 a바이트 읽기 명령이 내려진 경우, 각각의 DCU가 $a/8$바이트의 데이터를 MCU에게 보내고(부-PCI, 주-PCI), MCU에서 캐쉬로 그 데이터를 보낸 후에, HIU가 캐쉬를 엑세스한다(주-PCI). 결과적으로 MCU와 캐쉬, 캐쉬와 HIU 사이에는 a바이트가 전송된다.

주-PCI 버스와 부-PCI 버스에서 데이터를 전송하는 데에 걸리는 시간을 계산하면 다음과 같다.

$$\begin{aligned} T_{read,주-PCI,1} &= T_{MCU} + T_{HIU} + T_{DCU,1}(4)/8 \\ &= a/8 + a/8 + a/8 \\ &= 3a/8 \end{aligned}$$

$$\begin{aligned} T_{read,부-PCI,1} &= T_{DCU,1}(4)/8 \\ &= a/8 \end{aligned}$$

또한 전체 데이터 전송 소요 시간을 계산하면 다음과 같다.

$$\begin{aligned} T_{read,1} &= MAX[T_{read,주-PCI,1} + T_{read,부-PCI,1}] \\ &= 3a/8 \end{aligned}$$

여기서 MAX 함수가 사용된 것은 주-PCI와 부-PCI가 파이프라인 형식으로 동작하는 것으로 가정하였기 때문이다. 일반적으로 RAID에서는 많은 작업들이 계속해서 요청되므로 MCU에서 디스크 입출력을 하면서, 동시에 캐쉬를 거쳐 HIU로 데이터를 전달하는 경우가 많다. 결과를 분석해보면 주-PCI 버스가 부-PCI 버스에 비해서 3배나 많은 시간이 소요되어, 시스템 전체의 병목이 되는 것을 알 수 있다.

4.2 시스템 구성 2의 성능 분석

호스트에서 읽기 명령이 내려진 경우에는, DCU로 디스크 읽기 작업이 요청된다. 즉, 각각의 DCU가 a/8바이트의 데이터를 MCU에게 보내고(부-PCI, 이차-PCI), MCU에서 캐쉬로 그 데이터를 보낸 후에, HIU가 캐쉬를 엑세스한다(주-PCI).

주-PCI 버스와 부-PCI 버스에서 데이터를 전송하는 데에 걸리는 시간을 계산하면 다음과 같다.

$$\begin{aligned} T_{read,주-PCI,2} &= T_{MCU} + T_{HIU} \\ &= a/8 + a/8 \\ &= a/4 \end{aligned}$$

$$\begin{aligned} T_{read,부-PCI,2} &= MAX[T_{DCU,1}(4), T_{DCU,2}(4)]/8 \\ &= MAX[a, 3a/2]/8 \\ &= 3a/16 \end{aligned}$$

여기서 부-PCI 버스의 수식은 본 논문에서 가정한 브리지의 특성에 기인한 것이다. 본 논문에서는 Delayed write와 Posted write 기능을 가진 브리지를 가정한다. 그러므로 MCU가 브리지 이후의 DCU를 엑세스하는 것과 브리지 앞의 DCU를 엑세스하는 것은 완전하게 병렬로 수행될 수 있으며 그 때 최대의 성능을 낼 수 있다. 그러므로 MCU가 부-PCI 버스의 DCU를 엑세스하는 시간과 이차-PCI 버스의 DCU를 엑세스하는 시간 중의 큰 값이 전체 엑세스 시간을 결정하게 된다. 위의 식들을 바탕으로 전체 데이터 전송 소요시간을 계산하면 다음과 같다.

$$\begin{aligned} T_{read,2} &= MAX[T_{read,주-PCI,2} + T_{read,부-PCI,2}] \\ &= a/4 \end{aligned}$$

결과적으로 주-PCI 버스와 부-PCI 버스의 부하가 비슷하지만 주-PCI가 전체 시스템의 병목이 되고 있다.

4.3 성능비교

<표 1>은 앞 절에 산출한 식들을 정리한 것이다.

<표 1> 구성1과 구성2의 읽기 성능 비교

	소요시간	병목
구성 1	3a/8	주-PCI
구성 2	a/4	주-PCI

구성 1의 경우와 구성 2의 경우의 디스크 읽기에서는 주-PCI 버스가 병목이 되었다. 디스크 읽기의 경우에는 구성 2가 구성 1에 비해서 33% 우수한 성능을 보였다. 이 수치만 따지면 성능의 차이가 그리 크게 보이지 않으나, 일반적인 경우 전체 디스크 엑세스 중에서 읽기의 비율이 매우 높다는 점을 감안하면 전체적으로 구성 2가 구성 1 보다 월등한 성능을 보인다고 평가할 수 있다.

시뮬레이션을 위해서는 시뮬레이터의 검증(verification)과 모델의 검증(validation) 과정이 모두 이루어져야 한다. 모델의 검증을 위해서는 정확한 파라미터를 가지고 시뮬레이션하고 이것을 실세계(real world)와 비교, 분석하여야 한다. 두 가지 구성의 RAID 시스템을 DEVS 형식론을 이용하여 기술하고, 이산사건 시뮬레이션 언어중의 하나인 DEVSim++를 이용하여 시뮬레이션 한 결과와 비교해 보고자 한다. 현재 시뮬레이터의 검증은 끝났으며 모델의 검증과 이를 통한 성능 분석은 연구가 진행중이다.

5. 결 론

RAID 시스템은 대용량의 데이터를 저장하는 시스템으로서 앞으로도 발전 가능성이 매우 큰 분야이다. 본 논문에서는 하드웨어 특히 버스 부하의 관점에서 RAID 시스템의 성능을 분석하였다. 그래서 RAID 시스템의 구성 요소들간의 트래픽을 분석하여 몇 개의 범주로 나누고 그때 일정량의 데이터를 전송하는 데에 걸리는 시간을 분석하였다. 또 그것을 바탕으로 2개의 시스템 구성 예에 적용하여 시스템의 읽기 및 쓰기 요청에 대한 처리 시간을 계산하고 비교하였다. 분석 결과 같은 구성 요소들로 이루어진 시스템임에도 불구하고 33%의 성능 차이를 보였다.

참고문헌

1) Marc Farley, *Building Storage Networks*, McGraw-Hill, 2000.
2) *William Stallings, Computer Organization and Architecture*, 5th ED., Prentice Hall, 2000.
3) Shenze Chen and Don Towsley, "A Performance Evaluation of RAID architecture," IEEE Trans. Computers, Vol. 45, No. 10, 1996.
4) 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.
5) A. L. Narasimha Reddy and Prithviraj Banerjee, "An Evaluation of Multiple-Disk I/O Systems," IEEE Trans. Computers, Vol 38, No. 12, 1989.
6) Fibre Channel Standards, http://t11.org
7) Tom Shanley and Don Anderson, *PCI System Architecture*, Addison Wesley, 1995.