단일 디스크 입출력에서의 디스크 이질성 지원

김호진(Hojin Ghim)

한국과학기술원 전자전산학과 전산학전공 컴퓨터구조연구실

2004년 8월 20일

요약

본 보고서에서는 클러스터 상에서 다양한 종류의 디스크를 지원하도록 개선된 단일 디스크 입출력에 대해 설명한다. 단일 디스크 입출력은 리눅스의 커널 모듈의 형태로 제작되었으며 클러스터 내의 다양한 디스크를 하나의 큰 디스크 장치처럼 보이도록 해주는 역할을 한다. 또한 다양한 성능의 디스크가 존재할 때 모든 디스크의 성능을 최대한 활용하도록 한다.

1. 서론

클러스터 시스템은 저렴한 가격과 확장성으로 인해 큰 인기를 얻고 있다. 클러스터 시스템이 많이 쓰이게 되고 규모 또한 점차 커지면서 이질적인 하드웨어에 대한 지원이 필수적이게 되었다. 본 보고서에서는 이질적인 디스크를 보유한 클러스터 시스템에서 SDIO[1]가 어떻게 이질적인 디스크를 활용할 것인가에 대한 해결 방안을 모색한다.

SDIO는 클러스터 시스템의 각 노드의 디스크를 묶어 하나의 큰 디스크로 보이게 하는 소프트웨어이다. 기존의 SDIO는 모든 디스크를 동일한 성능과 동일한 용량의 디스크로 간주하고 있다.

디스크의 이질성은 다양한 각도에서 정의될 수 있지만 본 연구에서는 디스크의 성능과 용량을 이질성의 기준으로 삼는다. 또한 성능이나 용량이 다른 디스크가 두 종류 이상 존재할 때의 시스템 환경을 이질적인 디스크 환경이라 부른다.

본 연구에서는 이질성 문제를 해결하기 위해 디스크의 이질성에 따라 디스크 블락의 배치를 조정할 수 있는 알고리즘을 제시하고 이 알고리즘을 적용시킨 SDIO(SDIO-HETERO)를 설명한 후 실험에 의한 성능을 보인다.

2. 관련연구

가. AdaptRaid5

여러 개의 디스크를 활용하여 전체 디스크 시스템의 성능을 높이고 데이터 신뢰도를 높이기 위해 일반적으로 RAID[2] 기법을 사용한다. 이 중 RAID-5는 여러 개의 디스크 중 하나에 패리티를 저장하여 디스크 장애 발생시에 데이터를 복구할 수 있도록 하며, 큰 데이터의 읽기/쓰기, 작은 데이터의 읽기에서 높은 성능을 보인다. 기존의 RAID는 homogeneous한 disk configuration만을 고려한 것이다.

AdaptRaid5[3]는 널리 쓰이고 있는 RAID-5를 heterogeneous 디스크 환경에서 사용하도록 변형시킨 것이다. 따라서 RAID-5의 장점을 그대로 얻을 수 있으며, 다양한 용량의 디스크에서 용량 활용률을 최대화시킨다.

이를 위해 AdaptRaid5에서는 다음의 순서에 따라 데이터 블락을 여러 디스크에 배치한다.

1. 디스크 성능이 좋을수록 디스크 용량도 크다고 가정한다. 용량이 큰 디스크에 많은 데이터를 저장하고, 용량이 작은 디스크에 적은 데이터를 저장한다. 이에 따라 디스크 주소에 따라 스트라이프의 크기가 달라진다.
2. RAID-5는 작은 데이터의 쓰기 작업이 느리다는 고질적인 문제점을 가지고 있다. 이러한 작은 데이터 쓰기의 성능이 낮은 문제를 최소화시키기 위해 모든 스트라이프를 가장 큰 스트라이프의 약수가 되도록 줄인다. 이로써 파일 시스템 레벨에서 가장 큰 스트라이프 크기를 기준으로 디스크 작업을 수행함으로써 작은 데이터에 대한 기록 작업 횟수를 줄일 수 있다.
3. 스트라이프 크기가 줄어들어서 비게 된 디스크 블락에 다음 디스크 블락의 내용을 밀어 채운다. 이로써 빈 공간 없이 모든 디스크 공간을 활용할 수 있다. 디스크 공간 활용률은 거의 100%에 가깝게 된다.
4. 앞의 순서가 끝난 상태에서 용량이 작은 디스크는 주소 공간의 앞쪽에만 위치되게 된다. 따라서 전체 디스크 시스템의 주소 공간 중 앞쪽에 있는 파일은 큰 스트라이프를 사용해 모든 디스크에 다 저장되지만 주소 공간 중 뒤쪽에 위치한 파일은 작은 스트라이프를 사

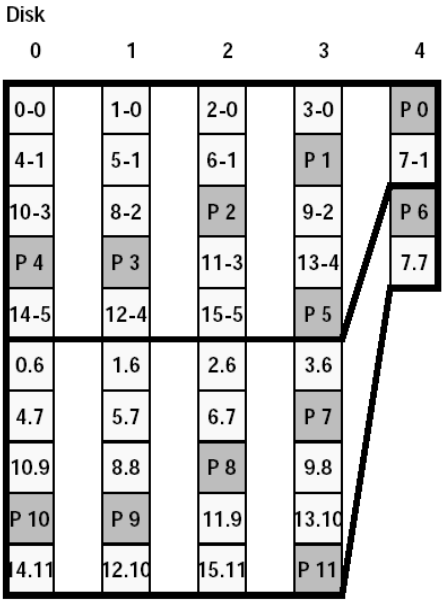

그림 1 AdaptRaid5의 디스크 블락 배치

용하므로 보다 작은 수의 디스크에 분포되게 된다. 이는 후자에 대해 병렬 작업의 효과를 낮춤으로써 비교적 낮은 성능을 나타내게 한다. 이 문제를 해결하기 위해 스트라이프를 큰 것부터 만들지 않고 크고 작은 스트라이프의 패턴을 반복시킨다. 이로써 파일이 주소 공간의 어느 위치에 있어도 큰 스트라이프와 작은 스트라이프가 골고루 사용되게 된다.

그림 1은 위 알고리즘의 결과를 보여준다.

AdaptRaid5는 위의 알고리즘을 사용함으로써 heterogeneous 디스크 환경의 모든 용량을 활용하면서 RAID-5와 비슷한 성능과 신뢰도를 보이고 있다. 그러나 단순한 계산으로 주소가 결정되는 RAID-5와 달리 addressing에 다중 레벨의 테이블을 필요로 하며 그 검색 알고리즘도 복잡해진다. 또한 디스크 블락 배치가 고정되어 disk configuration의 변동에 적응하기 어렵다.

나. Panda

Panda[4]는 대용량 과학 계산 어플리케이션을 위해 개발된 I/O 라이브러리이다. Panda는 collective I/O 형태의 API를 제공하며 원래 수퍼 컴퓨터에서 개발되었으나, 후에 클러스터 시스템에서도 사용할 수 있도록 개선되었다. collective I/O API를 통해 받는 데이터는 파일 시스템에서 보통 쓰이는 바이트 스트림이 아니고 다차원 배열의 데이터이다. Panda 라이브러리는 이 데이터를 I/O 노드에 적절히 분배하여 저장한다.

데이터가 디스크에 분배되는 과정은 다음과 같다.

1. I/O 서버 개수와 I/O 서버의 선택

I/O 서버의 수는 미리 정해져 있을 수도 있고 동적으로 결정할 수도 있다. 일단 모든 노드의 I/O 성능을 측정한 후 그 결과에 따라 전체 throughput의 합계가 가장 높을 것으로 예측되는 I/O 서버 수를 선택한다. 클러스터의 인터커넥션 네트웍의 속도가 낮을 경우에는 데이터의 locality를 고려하여 I/O 서버를 결정한다. 인터커넥션 네트웍의 속도가 높을 경우에는 I/O 서버의 load balancing을 고려하여 I/O 서버를 결정한다.

2. 데이터 나누기

데이터 나누기는 두 단계로 이루어진다. 첫 단계는 데이터의 액세스 패턴에 따라 사용자가 직접 지정해주는 것이다. 다음 단계는 사용자에 의해 나누어진 데이터를 작은 단위로 잘게 나누는 작업이다. 데이터를 좀더 잘게 나눔으로써 더욱 세밀하게 load balancing을 수행할 수 있다.

3. I/O 서버에 데이터 할당

이 작업은 1번 과정에서 측정된 I/O 서버의 성능에 따라 데이터를 배당하는 작업이다. 1번 과정에서의 성능을 기준으로 할당해놓고 계속 사용하는 static 방법이 있고, 매번 I/O가 발생할 때마다 지난 I/O에서의 성능을 기준으로 재할당하는 dynamic 방법이 있다. dynamic과 static을 적절히 조합한 hybrid 방법을 사용할 수도 있다.

Panda는 여러 노드에서 대용량의 데이터를 동시에 읽고 쓰는 collective I/O에 알맞게 만들어져 있다. collective I/O API를 사용하는 어플리케이션에서 아주 높은 성능을 보일 수 있다. 그러나 collective I/O는 블락 단위로 작업하는 디스크 레벨에서는 제공할 수 없는 API이다. 따라서 일반적인 사용에는 적합지 않다.

다. RIO

RIO[5](Randomized I/O)는 멀티미디어 데이터를 위한 스토리지 시스템이다. 여러 개의 디스크를 사용하여 멀티미디어 데이터 요구에 대한 동시 처리량을 높이고 또한 최대 지연 시간을 보장함으로써 실시간 전송을 지원하는 것을 목적으로 한다.

RIO는 원본 데이터와 복제본의 두 가지 측면에서 데이터 allocation을 수행한다.

원본 데이터는 random 디스크의 random 블락에 저장된다. uniform random 함수를 사용함으로써 모든 데이터의 long-term load balance가 보장된다. 그러나 randomization으로 short-term load balance는 제공되지 않는다. RIO에서는 short-term load balance를 제공하기 위해 복제본을 저장한다.

복제본의 저장 위치를 결정하기 위해 디스크의 성능을 나타내는 지표 중 하나인 BSR[6]을 사용한다. BSR은 Bandwidth-Space Ratio의 약자로서 용량에 비해 높은 성능을 보이는 디스크에서 높게 나타나는 척도이다. 원본 데이터를 BSR이 높은 디스크에 저장하고 복제본은 BSR이 낮은 디스크에 데이터의 일부분을 중복 저장하면 기본적인 로드는 원본 데이터에서 가져가고 부족한 밴드위쓰에 대해 복제본을 사용함으로써 목표하는 최대 밴드위쓰를 만족시키고, short-term load balance를 이룰 수 있다. 복제본이 기록된 낮은 BSR의 디스크는 비교적 용량이 크므로 많은 복제본이 기록되지만 각각의 복제본은 원본 데이터의 일부만을 저장하며 보조적인 역할을 하므로 디스크의 밴드위쓰를 초과하지 않을 수 있다.

시뮬레이션 결과 원본 데이터의 100% 복제본을 만들면 BSR이 높은 디스크에 복제본을 만드는 것과 확률적으로 같은 성능을 보였다.

RIO는 덩어리가 크고, 쓰기보다는 읽기가 주 작업인 데이터에 대해 좋은 성능을 보인다. 그러나 randomization 된 각각의 블락의 위치를 기록하기 위한 매핑 테이블의 크기가 필연적으로 커지게 되므로 성능의 장애 요소가 된다. 또한 작은 데이터나 메타데이터를 많이 사용

하는 파일 시스템 작업에서는 좋은 성능을 낼 수 없다. 두 가지 종류의 디스크만을 사용할 수 있으므로 heterogeneity의 지원이 제한적이다.

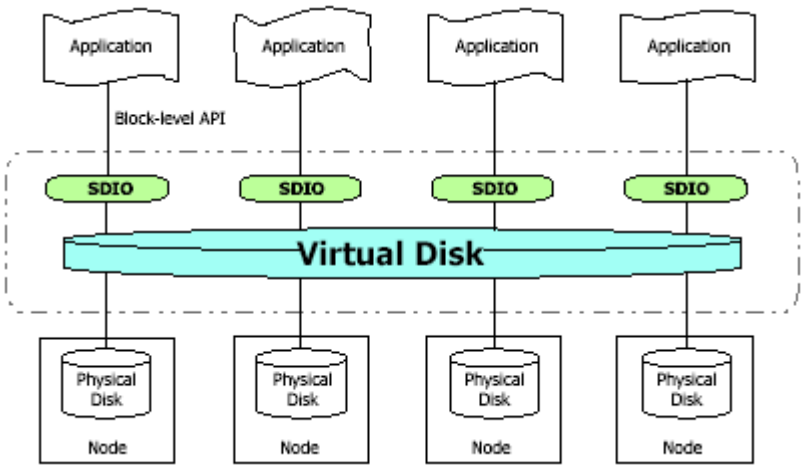

그림 2 SDIO 서비스 개념

3. Single Disk I/O (SDIO)

SDIO는 클러스터 환경에서 여러 노드에 장착되어 있는 디스크들을 하나의 디스크처럼 사용하도록 해주는 SSI 서비스의 일종이다. SDIO는 장치 드라이버 수준에서 구현되어 있으며 응용 프로그램에서 하나의 디스크를 사용하듯이 SDIO를 사용하면 실제 데이터는 클러스터 시스템에 흩어져 있는 디스크에 저장된다.

이처럼 하나의 가상 디스크를 제공함으로써 응용 프로그램은 데이터의 실제 위치에 신경쓸 필요없이 전체 클러스터 시스템의 디스크를 모두 사용할 수 있다. 또한 여러 개의 디스크를 병렬 작업에 활용하여 하나의 디스크에서보다 더욱 높은 성능을 얻을 수 있으며, 데이터를 중복 저장하여 필연적으로 발생할 수밖에 없는 디스크의 장애에도 불구하고 데이터의 유실을 방지할 수 있다.

SDIO는 장치 드라이버 수준에서 구현되어 있으므로 블락 단위의 Application Programming Interface(API)에서 투명성이 보장되므로 응용 프로그램이 보기에 물리적인 하드디스크와 완전히 같다. 따라서 SDIO에서 제공하는 가상 디스크에 일반적인 디스크에 사용되는 파일시스템을 수정 없이 사용하는 것이 가능하다. 파일시스템을 사용하는 응용 프로그램들도 마찬가지로 소스코드 수정이나 재 컴파일이 필요 없으므로 이진 코드 수준의 호환성이 제공된다.

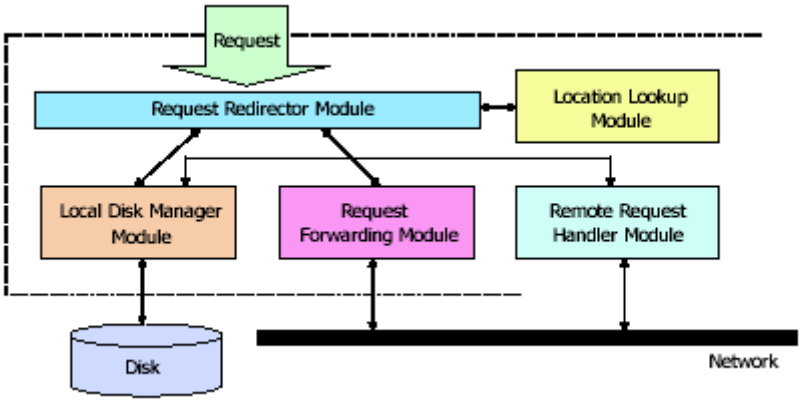

그림 3 SDIO 서비스 구성도

SDIO는 여러 노드에서 수행중인 여러 응용 프로그램에 의해 동시에 접근될 때의 일관성을 보장하지는 않는다. SDIO는 응용 프로그램에 블락 단위의 API를 제공하는데 블락 단위의 API는 파일 단위의 API와는 달리 한 블락에 대한 작업 요청이 이어지는 다른 작업 요청과 연관성이 없다. 여러 노드에서 여러 응용 프로그램의 동시 접근을 일관성 있게 허용하기 위해서 SDIO 위에 분산 파일시스템을 사용할 수 있다.

그림 3은 SDIO의 구성도이다. 이 그림은 클러스터 시스템의 각 노드에 적재되어 있는 장치 드라이버의 내부 구조를 나타낸 것이다.

SDIO 장치 드라이버로 들어오는 I/O 요청은 Request Redirector Module로 전달된다. 이 모듈은 Localtion Lookup Module을 사용하여 요청된 데이터 블락이 로컬 노드의 디스크에 있는지 확인한다. 데이터 블락이 로컬 노드의 디스크 안에 있을 때 Request Redirector Module은 Local Disk Manager Module에 디스크 작업을 요청한다. 데이터 블락이 로컬 노드의 디스크 안에 있지 않을 때 Request Redirector Module은 I/O 요청에 해당하는 내용을 Request Forwarding Module로 보낸다. 이 때 Localtion Lookup Module에서 알아낸 데이터 블락의 위치도 함께 보내진다. Request Forwarding Module은 네트웍을 통해 데이터 블락을 가지고 있는 노드로 I/O 요청을 전송한다. 네트웍을 통해 전송된 I/O 요청은 리모트 노드의 SDIO 장치 드라이버 내의 Remote Request Handler Module이 받는다. Remote Request Handler Module은 Local Disk Manager Module을 이용하여 디스크에

원하는 작업을 수행하고 난 뒤 다시 네트웍을 통해 결과값을 전송한다. Request Forwarding Module에서 결과값을 받으면 Request Redirector Module에서 응용 프로그램으로 결과를 되돌려준다.

각 데이터 블락이 어느 노드의 어느 디스크에 존재하는지는 Location Lookup Module에 의해 결정되어 있다. 현재 Location Lookup Module은 기존에 RAID 레벨 0과 RAID 레벨 1을 사용하여 제작되어 있고, 필요에 따라 새로운 방법을 사용할 수 있도록 유연하게 설계되어 있다.

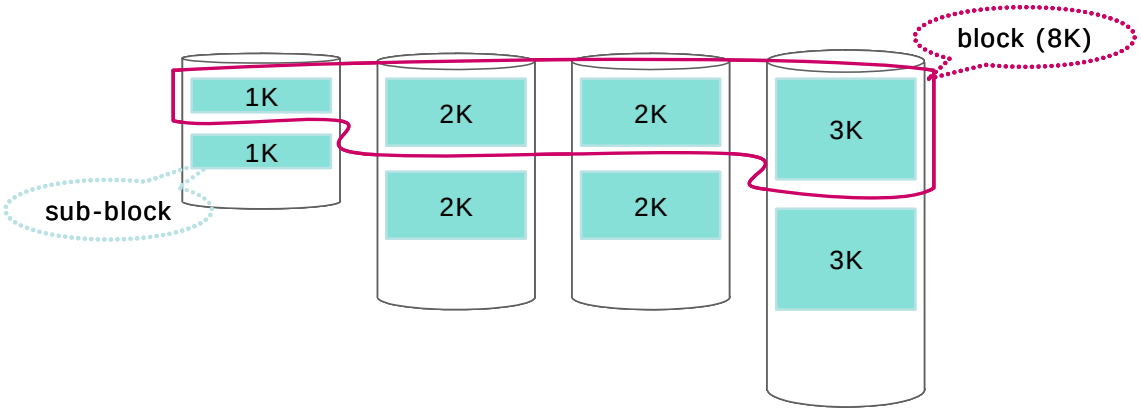

그림 4 디스크와 블락과 서브블락의 관계

4. SDIO-HETERO

가. 기본 알고리즘

SDIO-HETERO는 본 절에서 설명하는 알고리즘을 사용하여 디스크 블락의 배치를 결정한다. 디스크 블락의 배치는 시스템 초기 설치시에만 수행할 수 있다.

SDIO-HETERO에서 하나의 블락은 여러 개의 서브블락으로 구성되며, 각 서브블락은 서로 다른 노드의 다른 디스크에 존재할 수 있다. 모든 블락의 크기는 같지만 블락을 구성하는 서브블락의 크기와 수는 다를 수 있다. 서브블락의 크기는 디스크의 특성에 따라 결정되기 때문에, 한 디스크에 존재하는 서브블락의 크기는 모두 같다. 디스크와 블락, 서브블락의 관계를 그림 4에서 설명하고 있다.

모든 디스크 접근의 단위는 블락이고, 한 번의 블락 접근에 여러 개의 서브블락에 대한 접근이 동시에 일어나게 된다.

디스크 블락 배치 알고리즘은 각 블락이 어떤 서브블락으로 구성되며, 각 서브블락이 어느 노드의 어느 디스크에 위치하는지를 결정하는 과정이다.

1. 시스템에 포함된 모든 디스크에 대해 서브블락의 크기를 정한다. 모든 디스크에서 서브블락에 접근하는데 필요한 시간이 동일하도록 서브블락의 크기를 정한다. 디스크의 성능에 따라 데이터에의 접근 시간이 다르므로 빠른 디스크의 서브블락은 용량이 비교적 크고, 느린 디스크의 서브블락은 용량이 작아진다. 모든 서브블락에 대한 접근 시간이 같아

지므로 하나의 블락에 접근할 때 속도가 느린 디스크가 병목 현상을 발생시키지 않게 된다.

현재 대부분의 디스크의 최소 접근단위는 512바이트이므로 서브블락의 크기는 512바이트의 배수가 된다.

2. 서브블락들이 구성하는 블락은 SDIO-HETERO에서 기본 작업 단위가 된다. 일반 디스크 사용시 대부분의 작업 단위가 4KB이거나 그 약수이므로 기존 파일시스템이나 어플리케이션들은 4KB의 배수의 단위로 디스크에 접근하도록 최적화되어 있는 경우가 많다. 따라서 SDIO-HETERO에서도 블락을 4KB의 배수로 맞추면 기존 어플리케이션의 최적화 기법이 그대로 적용될 수 있다.

   블락의 크기를 4KB의 배수로 맞추기 위해 먼저 모든 디스크의 서브블락의 크기의 합계가 4KB의 배수가 되도록 한다. 서브블락의 크기는 다음과 같이 조정할 수 있다.

가. 모든 디스크의 서브블락을 두 배 한다.

나. 디스크 중 BSR이 가장 큰 디스크의 서브블락 크기를 한 단계(512바이트) 줄인다.

   BSR(Bandwidth-to-Space Ratio)은 디스크의 특성 중 하나로 제공하는 용량에 비해 성능이 얼마나 좋은지를 나타내는 지표이다. 서브블락의 크기를 줄이는 양은 디스크의 하드웨어적인 특성상 512바이트로 정해져 있는데 줄어드는 512에 따른 디스크 밴드위스 활용률의 감소는 디스크의 BSR에 따라 달라진다.

   1 단계를 수행하면 모든 서브블락은 접근 시간이 같게 된다. 여기서 한 서브블락의 크기를 줄이는 것은 해당 디스크가 다른 디스크보다 작업을 일찍 끝내고 다른 디스크의 작업이 끝나길 기다리는 결과를 가져온다. BSR이 가장 큰 디스크의 서브블락을 줄이면 줄어든 512바이트에 대한 작업 시간의 감소가 가장 작으므로 디스크 밴드위스 활용률이 가장 적게 줄어든다고 볼 수 있다.

3. 서브블락을 여러 개의 그룹으로 나눈다. 이 때 모든 그룹은 속한 서브블락의 크기의 합계가 같아야 한다. 한 그룹으로 묶인 서브블락들이 곧 하나의 블락을 이루게 된다. 같은 크기의 그룹으로 묶기가 불가능한 경우 2.가 단계를 한번 더 거치고 3 단계를 수행한다. 그림 5는 그룹으로 묶은 결과를 예시한 것이다.

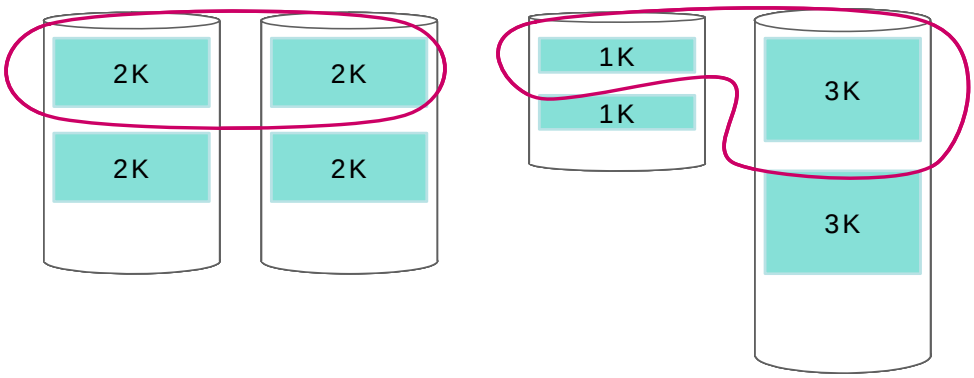

그림 5 서브블락을 그룹지어 하나의 블락을 만든다.

나. 구현

SDIO-HETERO의 구조는 SDIO와 전체적으로 같다. 그러나 하나의 블락 작업 수행시 여러 개의 서브 블락에 대한 접근을 위해 Request Redirector Module이 수정되었으며, 서브 블락의 위치를 찾기 위해 Location Lookup Module 또한 변경되었다.

Reqest Redirector Module은 한 번의 블락 작업에서 여러 서브블락에 대한 요청을 생성해서 Local Disk Manager Module이나 Request Forwarding Module에 전달한다.

Location Lookup Module은 블락에 속한 서브블락이 어느 디스크에 있는지를 저장하는 간단한 테이블을 생성함으로써 작업이 이루어질 디스크를 신속히 검색할 수 있다. 디스크 내에서의 서브블락의 위치는 블락 번호(B)와 그룹 수(G), 서브블락의 크기(s)를 사용하여 다음의 식으로 계산할 수 있다.

$$offset = \left\lfloor \frac{B}{G} \right\rfloor \times s$$

주어진 디스크의 서브블락 크기(s)가 몇인지도 하나의 작은 테이블에 저장해놓고 사용한다. 블락->디스크 테이블의 크기는 디스크 수이며, 서브블락 크기 테이블의 크기도 디스크 수이므로 매주 작은 메모리만을 차지하며 검색 또한 빠르다.

5. 성능 평가

성능 평가에 사용된 장비는 다음과 같다.

기종	A	B
CPU	Intel Pentium IV 1.8GHz	Intel Pentium III 850MHz
메모리	512MB	1.5GB
하드디스크 모델명	IBM Deskstar 120GXP	SAMSUNG SpinPoint 40GB
하드디스크 인터페이스	UATA100	UATA100
하드디스크 내장 버퍼	2MB	512KB
하드디스크 전송률	100MB/s	100MB/s
하드디스크 평균탐색시간	8.5ms	9ms

100메가바이트를 읽고 쓰는 간단한 실험 결과 위 기종의 디스크 작업 성능은 다음과 같이 나타났다.

기종	A	B
읽기	41323 KB/s	9051 KB/s
쓰기	29022 KB/s	6899 KB/s

위 결과에 따르면 A는 B에 비해 읽기에서 약 4.5배, 쓰기에서 약 4.2배의 성능을 보인다.

성능 평가는 데이터 읽기와 쓰기를 반복했을 때 읽기와 쓰기의 시간당 평균 데이터 전송률을 척도로 이루어졌다. 데이터에 대한 읽기와 쓰기는 일정한 레코드 크기의 데이터를 연속적으로 읽어서 총 읽은 양이 100메가가 될 때까지 진행하여 읽기의 시간당 전송률을 구하고, 같은 방법으로 쓰기의 시간당 전송률을 구한다. 읽기와 쓰기를 번갈아 4번 반복하여 처음 반복의 전송률은 버리고 나머지 세 번의 전송률의 평균을 계산하였다.

쓰기를 수행할 때 한 번의 레코드를 저장할 때마다 디스크에 동기화를 수행하여 메모리 캐시의 영향을 줄이도록 했다.

클러스터 시스템 구성은 다음의 네 가지를 사용하였다. A와 B는 사용된 노드 기종을 나타내고, 괄호 안의 숫자는 해당 디스크에 적용된 서브블락 크기를 뜻한다.(단위: KB)

아래 표 중 hetero가 SDIO-HETERO가 적용된 구성이다. 앞에서 나타난 A 기종과 B 기종의 성능 차에 따라 A 기종에는 B 기종의 네 배의 크기의 서브블락을 사용하도록 하였다.

small-B	hetero	small-A	large-A
B(4)	B(4)	A(4)	A(16)
B(4)	B(4)	A(4)	A(16)
B(4)	A(16)	A(4)	A(16)
B(4)	A(16)	A(4)	A(16)

그림 6은 읽기 작업에 있어서 레코드 크기가 달라짐에 따라 위의 네 가지 구성의 데이터 전송률을 나타내며, 그림 7은 쓰기 작업에 있어서 레코드 크기가 달라짐에 따라 위의 네 가

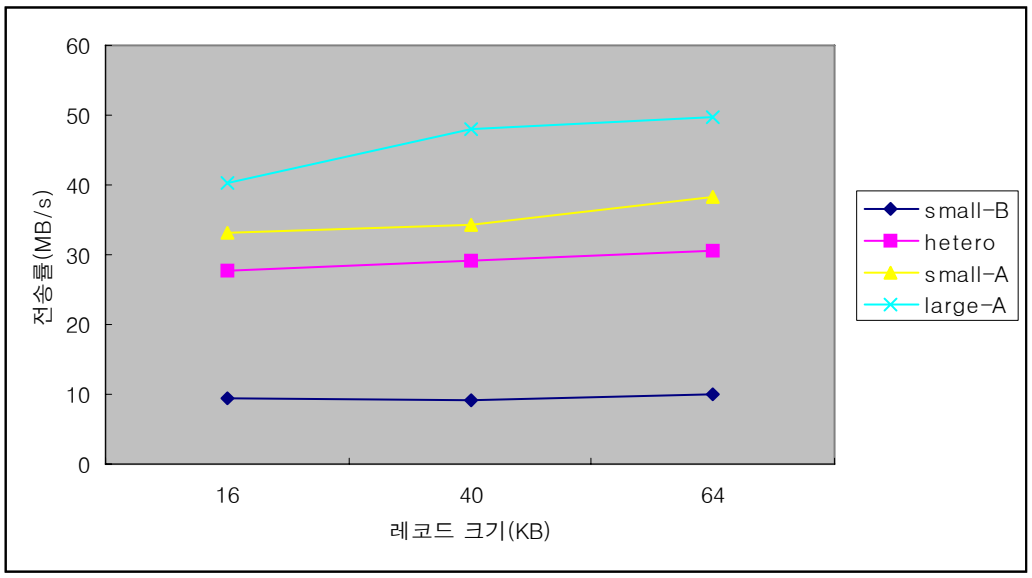

그림 6 읽기 작업에 대한 전송률

지 구성에서의 데이터 전송률을 나타낸다.

그래프에 보이는 바와 같이 SDIO의 성능은 구성 노드의 성능 뿐 아니라 서브블락의 크기, 레코드 크기에 따라서도 변화함을 알 수 있다. SDIO-HETERO는 느린 노드(B)와 그보다 네 배의 성능을 지닌 노드(A)가 반반 섞여 있어 빠른 노드로만 이루어진 구성과 느린 노드로만 이루어진 구성의 중간이거나 중간을 약간 상회하는 성능을 보임을 알 수 있다. SDIO-HETERO의 다른 구성에 대한 성능 비율은 레코드 크기가 작을 때 더 높음을 볼 수 있는데 이는 레코드의 크기가 클 때 서로 다른 크기의 서브블락의 경계를 넘어서는 경우가 많아지기 때문인 것으로 이해할 수 있다.

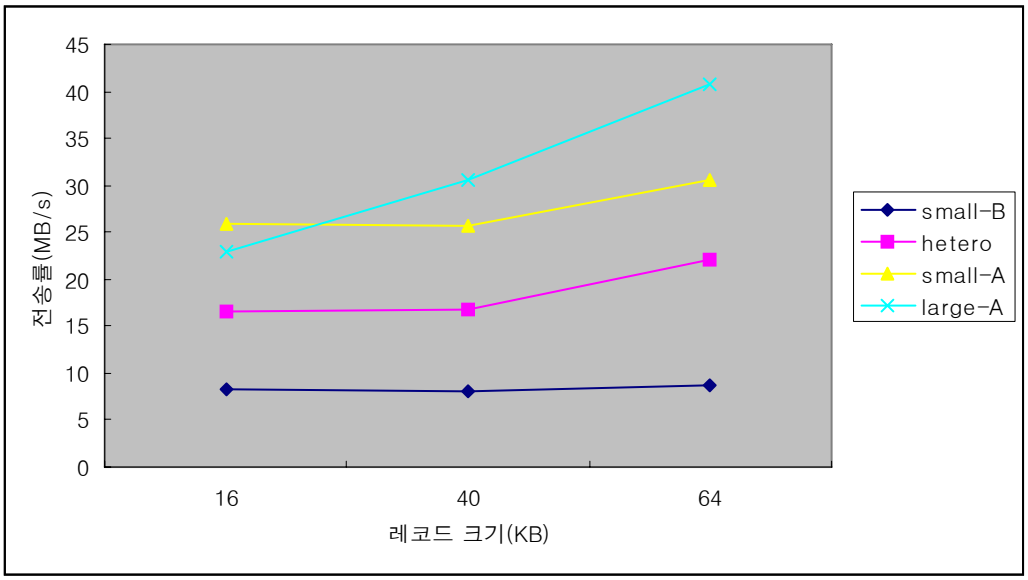

그림 7 쓰기 작업에 대한 전송률

6. 결론

클러스터 시스템은 규모가 커지고 유지보수가 진행될수록 동질 노드만으로 구성하는데 큰 비용을 필요로 하게 된다. 본 연구는 이러한 기존 클러스터 시스템을 완전히 대체하지 않고 새로운 구성품만을 추가함으로써 합리적인 비용에 성능을 높일 수 있는 가능성을 제시한다. 이 보고서에서는 이질적인 디스크에 블락을 적절히 분배할 수 있는 알고리즘을 제시한다. 또한 제시된 알고리즘을 구현하는데 관련된 사항을 다루었으며, 구현된 SDIO-HETERO의 성능을 실험을 통해 확인하고 있다.

SDIO-HETERO는 디스크의 가능한 이질성 중 성능을 중점적으로 처리하고 있다. 이에 따라 모든 디스크의 용량을 전부 사용하지는 못하고 있다. 따라서 이렇게 남는 디스크 용량을 성능에 영향을 주지 않는 한에서 유용하게 사용하는 방법을 찾는 것이 차후 과제로서 남겨져 있다.

7. 참고문헌

[1] 황인철, 김동환, 김호진, 맹승렬, 조정완, “단일 디스크 입출력을 위한 커널 모듈 프로토타입의 설계 및 구현”, 한국정보과학회 2003년도 추계학술발표논문집, 2003

[2] D. A. Patterson, G. A. Gibson and R. Katz, "A Case for Redundant Arrays of Inexpensive Disks", Proceedings of ACM SIGMOD '88, pp109-116, 1988

[3] T. Cortes and J. Labarta, "Extending Heterogeneity to RAID level 5", Proceedings of the General Track: 2001 USENIX Annual Technical Conference, pp119-132, 2001

[4] Y. E. Cho, M. Winslett, S. Kuo, J. Lee and Y. Chen, "Parallel I/O for Scientific Applications on Heterogeneous Clusters: a Resource-Utilization Approach", Proceedings of the 13th International Conference on Supercomputing, pp253-259, 1999

[5] J. R. Santos and R. Muntz, "Performance Analysis of the RIO Multimedia Storage System with Heterogeneous Disk Configurations", Proceedings of the 6th ACM International Conference on Multimedia, pp303-308, 1998

[6] A. Dan and D. Sitaram, "An Online Video Placement Policy Based on Bandwidth to Space Ratio (BSR)", Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, pp376-385, 1995