리눅스 고가용 시스템에서 로컬 디스크 간 데이터 동기화 구현

Implementation of data synchronization for local disks in Linux high availability system

박성종, 이철훈*
충남대학교 컴퓨터공학과

Park seong-jong, Lee cheol-hoon*
Chungnam National Univ., Computer Engineering, System Software Laboratory

요약

최근에 블로그, UCC, IPTV 등 사용자 중심의 인터넷 서비스와 언제 어디서나 웹을 통해 서비스를 받을 수 있는 유비쿼터스 컴퓨팅 환경으로의 변화는 안정된 서비스를 제공할 수 있는 고가용 시스템 플랫폼을 필요로 한다. 고가용 시스템이란 네트워크상에 서버들을 클러스터로 구성함으로써 만약 서비스 하던 시스템이 고장과 같은 시스템 장애가 발생하더라도 계속해서 안전하게 서비스를 제공할 수 있는 시스템을 말한다. 그리고 이러한 고가용 시스템 플랫폼에서 서비스의 신뢰성을 위해 시스템 간 데이터 동기화는 필수적이다. 본 논문에서는 리눅스 고가용 시스템에서 로컬 디스크 간 실시간 데이터 동기화 기술인 DRBD(Distributed Replicated Block Disk)를 구현하였다.

Abstract

Recently, changes in the environment of user-centric internet service such as blog, UCC and IPTV and ubiquitous computing based on web service are needed to high availability system platform. High availability system is to provide safe service continuously even if system failure occurs in clustering system at the network. And it is necessary to synchronize data for reliable service in high availability system. In this paper, I implement DRBD(Disk Replicated Block Device) which is synchronization technique for data of local disks in high availability system.

I. 서 론

인터넷의 급속한 발전으로 세계 어느 곳에 있든지 네트워크 환경에 있다면 다양한 업무들을 처리할 수 있게 됐다. 전 세계가 인터넷이라는 거대한 하나의 고리로 연결됐을 뿐 아니라 인터넷 뱅킹, 전자 결제, 전자 상거래, 증권 거래, 민원 업무 등 우리 일상도 인터넷을 이용해 처리할 수 있게 된 것이다. 이처럼 인터넷이 생활 속으로 깊숙이 침투하고 이러한 서비스를 이용하는 사용자들이 빠르게 증가함에 따라 중단 없는 서비스에 대한 요구도 점점 높아지고 있다. 따라서 365일 항상 서비스를 제공할 수 있는 고가용 시스템이 절실하게 필요해졌다.

고가용 시스템은 네트워크상에 서버들을 클러스터로 구성함으로써 만약 서비스 하던 시스템이 고장과 같은 시스템 장애가 발생하더라도 계속해서 안전하게 서비스를 제공할 수 있는 신뢰성과 가용성을 보장 한다. 이를 위해 로컬 시스템간의 데이터 동기화는 필수적이다.

본 논문에서는 Active-Standby 방식의 리눅스 고가용 시스템에서 각 로컬 디스크 간 데이터 동기화를 위해 DRDB를 구현하였다. 2장에서는 관련연구로 고가용 시스템과 Heartbeat에 대하여 알아보고, 3장에서는 리눅스 고가용 시스템에서 로컬 디스크 간 데이터 동기화 구현, 4장에서는 실험환경 및 결과를 마지막으로 5장에서는 결론을 맺는다.

Ⅱ. 관련연구

1. 고가용 시스템

고가용 시스템은 하드웨어나 소프트웨어 또는 네트워크 등의 자원에 결함이 발생했음에도 불구하고 지속적으로 서비스를 제공할 수 있다. 즉, 결함과 시스템의 다운타임을 최소화함으로써 서비스의 손실을 줄일 수 있도록 컴퓨터 시스템을 구축하고 관리하는 시스템을 말한다.

최근 몇 년 동안, 개인용 PC에서 사용되던 인텔이나 AMD와 같은 업체들의 CPU 성능도 급격히 향상됐고, 리눅스와 같은 공개 소스 운영체제 들을 아주 저렴한 가격에 얻을 수 있게 되었으며, 100Mbps, 1Gbps 등의 고속의 네트워크를 구현하기 위한 표준과 하드웨어 제품군이 등장하여, 이들을 조합해 구축함으로써 저렴한 가격으로 고가용 시스템을 구축할 수 있게 되었다[1].

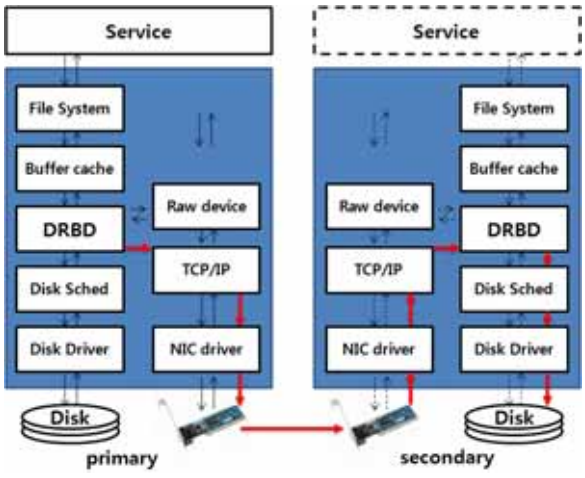
2. Heartbeat

Heartbeat은 'The High Availability Linux Project'에서 제공하는 고가용 소프트웨어로써 클러스터의 노드들은 서로의 상태를 알리기 위해 주기적으로 heartbeat 메시지를 교환한다. 메시지 전송 주기는 클러스터의 관리자가 설정할 수 있다. 고가용 소프트웨어의 주기적인 상태 메시지 이외에도 노드가 현재 실행중인 고가용 서비스에 대한 정보, 메시지의 재전송 요구, 클러스터 구성 정보 등의 교환을 위하여 메시지를 송수신 한다. heartbeat 메시지 전송을 위한 통신 채널이 SPOF(Single Point of Failure) 장애가 되는 것을 방지하기 위하여 메시지는 공용 및 전용 이더넷과 시리얼 케이블을 통해 전달할 수 있다. 이더넷을 통해 전송되는 메시지는 UDP/IP 브로드캐스트 프로토콜을 따라 클러스터의 모든 노드에게 효율적으로 전달된다[2].

Ⅲ. 로컬 디스크 간 데이터 동기화 구현

본 논문은 리눅스 고가용 시스템에서 로컬 디스크 간 동기화를 위해 DRBD를 구현하였다. DRBD는 고가용 시스템을 위해 설계된 블록 디바이스이다. 네트워크를 통해 블록 디바이스 전체를 미러링 하는데 이는 RAID-1 구성을 네트워크상에 구현한 것이다.

DRBD는 그림 1과 같이 로컬 디스크에 데이터를 쓰면 네트워크를 통해 다른 시스템에 그 데이터를 실시간으로 전송한다. 그럼 데이터를 받는 시스템은 데이터를 보낸 시스템과 동일한 데이터를 자신의 로컬 디스크에 가지게 된다.

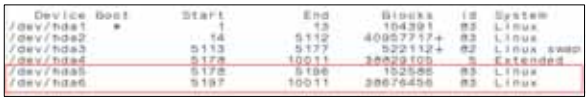


▶▶ 그림 1. Linux I/O stack에서 DRBD의 위치

각 DRBD는 'primary'나 'secondary' 상태를 가지고 있다. primary 상태를 가지고 있는 시스템의 응용프로그램은 디바이스를 접근하고 실행할 수 있다. 그리고 모든 파일 쓰기는 로컬 로우 레벨 블록 디바이스에 쓰고, secondary 상태의 시스템에게 데이터를 보낸다. secondary 시스템은 간단하게 로우 레벨 디바이스에 데이터를 쓰며 데이터 읽기는 항상 로컬 시스템에서 지역적으로 수행된다.

만약 primary 시스템이 고장 났을 경우 DRBD의 상태가 secondary였던 시스템은 primary 상태로 전환된다. 고장 난 시스템이 다시 정상으로 돌아오면 새로운 secondary 시스템이 되고 primary 시스템의 데이터를 동기화 한다.

그림 2와 같이 로컬 디스크에서 실제 공유하기 위한 파일이 저장될 DRBD 파티션과 DRBD에 데이터 보관과 관련된 다양한 정보들이 저장될 메타 데이터를 위한 파티션을 별도로 둔다.



▶▶ 그림 2. DRBD와 메타데이터를 위한 파티션

그림 3은 DRBD의 환경설정 파일인데 크게 세부분으로 나눠진다. global 섹션은 환경설정 파일의 가장 상위에 있는 부분으로 대부분의 사용자들은 한번 참조하는데, 섹션안의 usage-count 필드는 새로운 버전에 대한 업그레이드 진행 여부에 관련된 필드이다. common 섹션은 모든 자원에 적용할 프로토콜 방법을 제공한다.

```
global {
  usage-count yes;
}
common {
  protocol C;
}
resource r0 {
  on sslab1 {
    device     /dev/drbd0;
    disk       /dev/hdc6;
    address    192.168.0.10:7788;
    meta-disk  /dev/hdc5[0];
  }
  on sslab2 {
    device     /dev/drbd0;
    disk       /dev/hda6;
    address    192.168.0.11:7788;
    meta-disk  /dev/hda[5];
  }
}
```

▶▶ 그림 3. drbd.conf

프로토콜은 세 개로 나눠지는데 프로토콜에 대한 내용은 표 1과 같다. 구축된 고가용 시스템에서 시스템 다운타임시 최소한의 데이터 손실을 위해 프로토콜 C를 사용함으로써 각 시스템이 데이터를 동기화하도록 구현하였다.

[표 1] DRBD wirte protocol

protocol	설명
A	비동기화 프로토콜로써 Write I/O 처리에 대해 만약 데이터가 로컬디스크에 저장되고, 동기화할 데이터가 로컬 TCP Send Buffer에 저장되었을 때 완료된다.
B	비동기화와 동기화의 중간단계의 프로토콜로써 Write I/O 처리에 대해 만약 데이터가 로컬디스크에 저장되고, 동기화 할 데이터가 백업 시스템의 Buffer Cache에 저장되었을 때 완료된다.
C	동기화 프로토콜로써 Write I/O 처리에 대해 만약 데이터가 로컬디스크에 저장되고, 동기화 할 데이터가 secondary 시스템의 로컬 디스크에 저장되었을 때 완료된다.

마지막으로 resource 섹션은 DRBD의 자원에 관한 내용을 가지고 있는데 실제 디스크, 디바이스, 메타데이터가 저장될 파티션, 데이터 동기화할 때 사용되는 링크와 포트에 대한 정보를 가지고 있다. 그림 4와 같이 7788포트를 사용하기 위해서는 /etc/sysconfig/iptables에서 방화벽 설정을 변경해줘야 한다.

▶▶ 그림 4. DRBD를 위한 리눅스 방화벽 설정

'service drbd start'라는 명령어로 DRBD를 시작하고 'cat /proc/drbd' 명령어로 DRBD의 구현 상태정보를 확인할 수 있다. 확인 정보는 아래 표 2와 같다.

[표 2] DRBD 상태정보

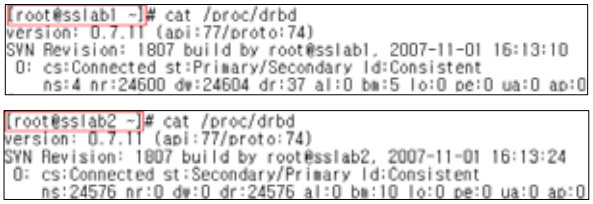
구분	설명
cs(connection state)	네트워크 연결 상태
st(states)	노드의 상태
ns(network send)	네트워크 연결을 통해 다른 시스템에 보내진 데이터의 크기
nr(network receive)	네트워크 연결을 통해 다른 시스템에서 받은 데이터 크기
dw(disk write)	로컬 디스크에 쓴 데이터 크기
dr(disk read)	로컬 디스크로부터 읽은 데이터 크기
al(activity log)	메타데이터 activity log 영역에 업데이트 된 횟수
bm(bit map)	메타데이터 bitmap 영역에 업데이트 된 횟수
lo(local count)	DRBD가 로컬 I/O 서브시스템에 열기 요청을 한 횟수
pe(pending)	다른 시스템에서 보낸 요청에 대해 아직 답변을 못한 횟수
ua(unacknowleged)	네트워크 연결을 통해 다른 시스템으로 받은 요청에 아직 답하지 못한 횟수
ap (application pending)	DRBD에게 전달되었지만 아직 답변을 받지 못한 블록 I/O 요청 횟수

Ⅳ. 실험환경 및 결과

본 논문에서는 리눅스가 설치된 PC 두 대에 고가용 기능을 제공하는 소프트웨어인 Heartbeat을 설치해 Active-Standby 고가용 시스템을 구축하고, 구축한 시스템에서 로컬 디스크 간 디스크 동기화 기능을 확인하기 위해 DRBD 구현과 리눅스 파일시스템에 DRBD 파티션을 만들었다[3-5]. 디스크 동기화를 위한 별도의 DRBD 전용 링크를 두어 큰 사이즈의 파일을 동기화할 때 고가용 시스템 네트워크 환경의 과부하를 덜어주었다.

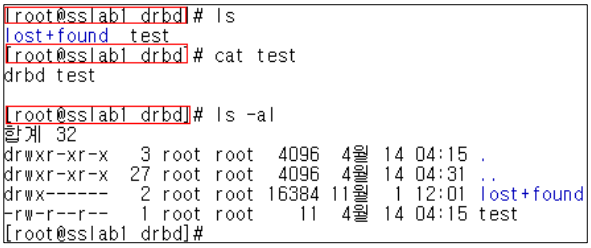
▶▶ 그림 5. 실험환경

① DRBD를 실행시키고 각 시스템의 DRBD 상태 정보를 확인한다. DRBD를 각 시스템에서 실행하면 그림 6과 같이 sslab1은 primary, sslab2는 secondary 상태가 된다.



▶▶ 그림 6. drbd 실행후 sslab1과 sslab2의 상태정보

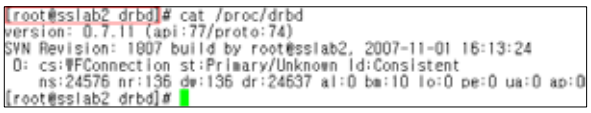
② primary 시스템인 sslab1의 /drbd 파티션에 파일을 만들고 내용을 확인한다.



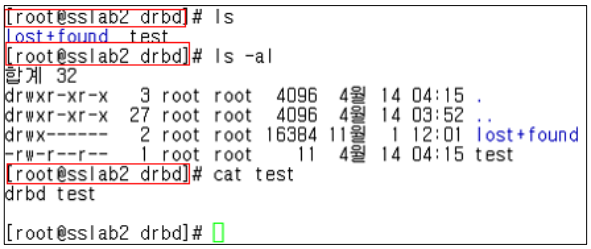
▶▶ 그림 7. sslab1 test 파일정보

③ primary 시스템의 DRBD 파티션에 파일을 만들면 실시간으로 네트워크를 통해 standby 시스템의 DRBD 파티션에 똑같은 파일이 생성된다.

primary 시스템인 sslab1을 정지시키면 standby 시스템 이였던 sslab2 상태는 primary 상태가 된다. 그림 9와 같이 sslab1의 test 파일은 sslab2의 /drbd 파티션에 저장되고 sslab1에서 저장했던 파일내용과 sslab2에 저장된 파일내용이 같음을 확인할 수 있다.



▶▶ 그림 8. sslab2의 상태정보



▶▶ 그림 9. sslab2 test 파일정보

V. 결 론

본 논문에서는 고가용 시스템에서 서비스의 신뢰성과 가용성을 높이기 위해 각 시스템 로컬 디스크 간 동기화를 구현하였다. 범용 운영체제인 리눅스에 고가용성 소프트웨어 Heartbeat을 설치한 시스템에서 동기화 할 파티션을 설정해 데이터를 쓰고 다른 시스템 같은 데이터를 확인함으로써 로컬 디스크 간 데이터 동기화를 확인하였다.

향후 연구 과제로는 리눅스 고가용 시스템에서 로컬 디스크 간 데이터 동기화의 성능 평가를 하는 것이다.

▌참 고 문 헌▐

[1] 김진억 "업무 연속성의 향상을 위한 고가용성 시스템 설계", 中央大學校 情報大學院 碩師學位論文, 2005.
[2] http://www.linux-ha.org/
[3] 김용희, 이재규, 박희상, 송대기, 이철훈 "Roll - Forward Recovery 를 적용한 고가용 웹 서버 시스템 구현", 2002 한국정보과학회 가을 학술발표 논문집 Vol. 29, No. 2, pp. 403-405
[4] 김용희, 성영락, 이철훈 "Cold-standby 방식의 고가용 웹 서버 시스템 구현", 2003 한국정보과학회 봄 학술발표 논문집 Vol. 30, No. 1, pp. 54-66
[5] http://www.drbd.org/