리눅스의 이중화 모듈 설계

정 상 진, 한 상 숙, 은 성 배
한남대학교 정보통신공학과
전화 : 042-629-8012 / 핸드폰 : 016-424-2373

a Design of Duplication Module in Linux

Abstract

In this paper, we have studied a design method to make Linux system enriched with a duplication facility. We have designed a duplication module which would be inserted into Linux. The design have to keep track of three foundations, such as fault detection, reconfiguration, a handling duplication data.

To conclude, Linux duplication module is free put on a Linux kernel. so in the future, It can cope with a change in Linux duplication system flexiblely.

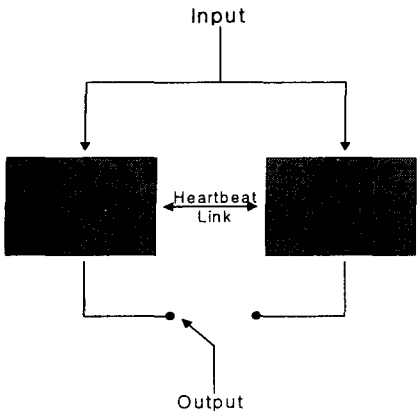
I. 서론

이중화 시스템은 정보의 교환을 담당하는 통신망 및 산업 시스템 전반에 걸쳐 결함 허용성을 적용하여 중단 없는 서비스를 제공하기 위한 시스템이다. 오늘날 통신 시장은 급속도의 발전을 거듭하며 낮은 가격, 질 높은 서비스를 제공하고 있으며 인터넷의 확산과 함께 학교, 회사, 연구기관을 벗어나 홈 네트워(Home Network) 시대로 가고 있다. 네트워의 확산은 정보화 시대에서 통신망을 이용한 정보 교환의 주요한 수단으로써 더욱 중요한 역할을 담당하게 될 것이다. 이때 이중화 시스템은 정보 교환의 주요한 수단이 되는 통신망 및 시스템에 다양한 서비스를 중단 없이 받을 수 있는 안전성 있고 신뢰 있는 서비스 시스템을 제공한다. 대부분의 이중화 시스템은 단일·멀티 NT 시스템과 UNIX 시스템등과 같은 서버급 컴퓨터에서 사용되고 있으며 비용도 가용성(Availability)에 따라 많은 차이가 나긴하지만 대부분은 이중화 시스템 위해 고가의 장비를 사용하고 있는 실정이다.[1]

본 논문에서는 최근 리눅스 시스템이 과거에 비해 많이 사용됨에 따라 전문적 관리 시스템이 아니더라도 주변의 리눅스 시스템을 이용하여 손쉽게 이중화 시스템으로 구축 할 수 있는 방안을 제시하려 하려하며, 해결책으로 이중화를 위한 리눅스 모듈을 만들어 손쉽게 이중화 시스템을 구축하는 방법을 제시하고자 한다. 본 논문에서는 리눅스 이중화를 위한 3가지 관점에 집중하여 리눅스상에서의 이중화 모듈 설계에 대해 연구한다. 논문의 구성은, 2절에서는 이중화 배경에 대해 알아보며, 3절에서는 리눅스 이중화 모듈 설계에 관하여 설명하며, 마지막으로 4절에서 결론을 맺는다.

II. 이중화 배경

고장 허용을 달성하기 위한 기술적 디자인은 하드웨어 중복, 정보 중복, 시간 중복, 소프트웨어 중복이 있다. 그 중에서도 하드웨어 중복 기능을 구현하기 위한 기본적인 방법으로는 아래의 그림[1]과 같은 standby sparing을 들 수 있다.[2]



[그림1] standby sparing 구조

2개의 모듈이 동시에 같은 입력 데이터를 받아 동시에 데이터를 처리를 하며, Active 모듈의 결과를 출력하게 된다. 만일 Active 모듈의 고장시 standby 모듈이 즉시 이를 탐지 하여 자신의 결과를 출력하는 간단한 구조로 되어 있다.

이중화 구조는 hot-standby, warm-standby, cold-standby 등의 여러 방법이 있지만 그 특성에 따라 다른 시스템에서 사용된다. cold-standby의 경우 한쪽 모듈이 동작하는 동안 다른 모듈은 정지 되어있으며 고장시 스페어 모듈이 처음부터 다시 시작하는 구조로 되어있는데, 전원을 절약해야 하는 시스템에 적합하다. warm-standby 구조는 cold-standby와는 다르게, standby 측에서 아무것도 하지 않는 것이 아니라 이중화를 위한 서비스를 지원하며 Active 모듈의 고장 시 이를 감지하여 Active 모듈의 서비스를 처리한다. 이것은 cold-standby보다 빠른 고장 처리를 지원한다. hot-standby 구조는 똑같은 2대의 서버를 사용하므로 장애 발생시 안정적으로 데이터를 처리 할 수 있으며, 장애 발생 탐지가 빠르며 standby로의 전환이 빨라 즉각 적인 서비스 제공이 가능하다. 그러나 디스크의 동기화와 마스터 서버의 감시 때문에 상당한 트래픽이 발생 하며 똑같은 서버 2대분 이상의 하드웨어 비용을 투자 해야 한다는 단점을 같고 있다.[3]

이중화 구조의 선택은 서비스를 하려는 내용과 구현 비용을 고려하여 효과적인 이중화 시스템을 구축 하여야 하는데, 본 논문서는 리눅스 이중화를 위해 hot-standby 보다는 느리지만 비용과 서버 활용도 면을 고려하여 warm-standby 구조를 사용 할 것이다.

시스템 이중화시 중요한 고려 사항은 다음과 같이 3가지로 나누어 볼 수 있다.

① 고장 감지
② 고장에 따른 재구성
③ 이중화 데이터 처리

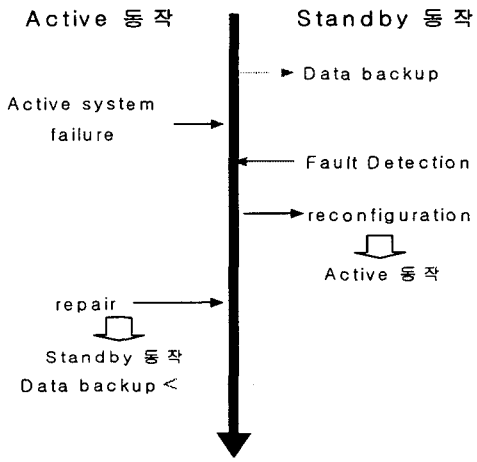
▶ 고장 감지

이중화를 위한 처음 단계는 Standby모듈에서 Active 모듈의 고장을 감지해 내는 것이다. 고장을 감지하는 방법에는 여러 가지가 있다. [그림1]에서 보는 것과 같이 전용 heart-bit 채널이나 워치독 타이머를 두어 이들의 연결성을 주기적으로 확인하여 장애 발생을 감지하거나, 출력 데이터를 조사 비교하여 잘못 된 데이터를 찾는 방법이 Standby 측에서 고장 감지를 위해 주로 사용 하고 있는 방법이다. 그밖에 Active 측에서도 스스템 장애를 발견 할 경우 능동적으로 Standby 모듈에 알려 고장을 처리하는 방법도 있다.

▶ 재구성

이중화 시스템은 두 시스템 모두 독립적인 상태에서 시작하며, 상대 시스템을 감지하여 상대가 Active이면 Standby로 아니면 Active로 전이한다.

하드웨어나 소프트웨어의 Active 모듈의 고장시 Active와 Standby 모듈간의 재구성이 이루어 져야 한다. 아래의 [그림2]는 Active와 Standby 모듈간의 재구성 시나리오를 보여주는 그림이다.[4]



[그림2] Active & Standby 모듈의 재구성

Active와 Standby가 동작하는 상황에서 Active는 관리자가 지정해둔 분기 때마다 지속적으로 데이터를 Standby 측에 백업을 하며, Standby는 Active의 상태를 지속적으로 검사한다. 만일 Active 모듈의 고장 발견시 Standby는 Active로 전이되며 백업해둔 데이터를 가지고 기존의 Active 모듈의 작업을 계속 수행한다. 고장된 기존의 Active 모듈은 수리 후 재 접속시

Standby 모듈로 동작하며, 각 모듈은 상황에 맞게 작업을 계속 수행하게 된다.

▶ 이중화 데이터 처리

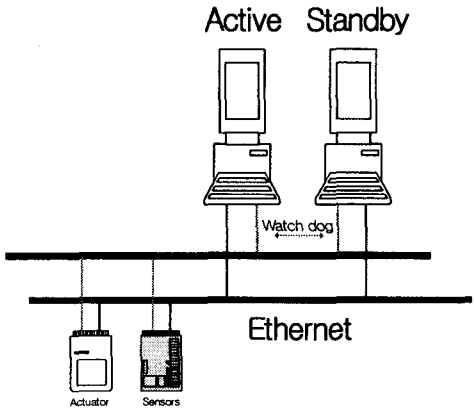
이중화시 데이터 처리는 크게 Tightly Coupled와 Loosely Coupled로 나누어 볼 수 있다. Tightly Coupled 방법은 데이터 저장이 필요 할 때 2개의 똑같은 하드에 동시에 데이터를 저장하는 방법이며 그 예로 TDX-10 교환기가 있다. Loosely Coupled 방법은 하나의 하드에 Active와 Standby 두 시스템이 연결되어 두 시스템 중 먼저 활성화된 시스템이 마운트하여 데이터를 저장하며 고장시 Standby 시스템이 이를 마운트하여 계속 작업을 수행하는 방법이며, 그 예로 AXE-10 교환기가 있다.[5]

이상 3가지 사항을 기본으로 이중화가 이루어지며, 본 논문에서도 3가지 기본사항에 맞추어 다음 절에서 리눅스 모듈을 설계한다.

III. 리눅스의 이중화 모듈 설계

3.1 리눅스 이중화 구조

아래의 [그림3]는 본 논문에서 최종 목표로 구현할 이더넷 상의 리눅스 시스템의 이중화 구성을 나타내는데, 그 특징은 이덧넷을 통하여 데이터의 전송과 고장을 감지하기 위한 워치독 타이머 수행이다.



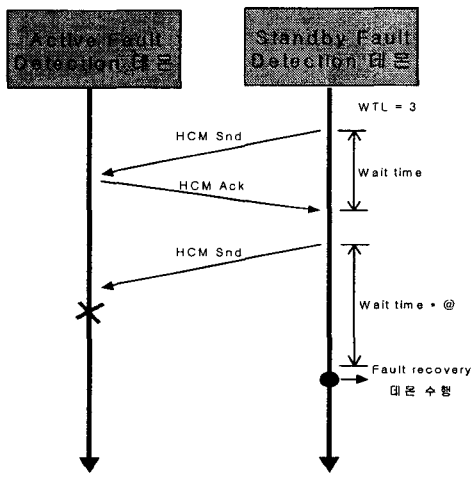
[그림3] 이더넷 상의 리눅스 이중화 연결구조

이중화된 이더넷을 통해 입력된 데이터는 Active 측에서 서비스를 처리하며, Standby 측에서는 아래의 [그림3]과 같이 워치독 타이머를 사용해 Active와 Standby 모듈간의 주기적 이벤트 메세지를 받아 고장을 감지한다. Active는 Standby에 주기적인 백업 데이터를 보내며, Standby는 Active에 hearbeat 체크 메세지를 보내어 고장 유무를 확인 한다. 이러한 일련의 수행들은 각각의 서비스 데몬들이 처리 한다.

3.2 이중화 모듈 설계 및 제어

리눅스 이중화 모듈 설계는 이중화 고려사항에 따라 3가지로 볼 수 있다.

첫 번째, 본 논문에서는 고장 감지를 위해 워치독 타이머를 사용해 주기적으로 Standby 측에서 Active 측으로 Hearbeat 체크 메세지를 보내어 고장을 감시한다.

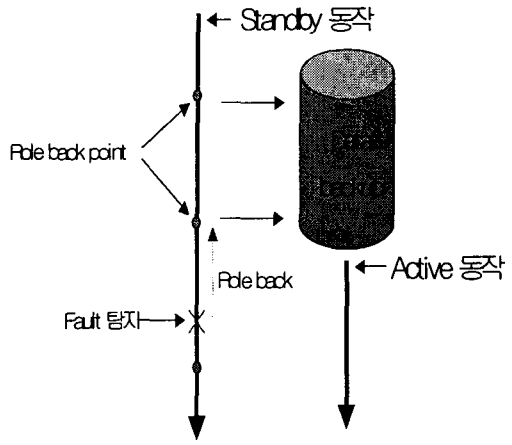


[그림4] 고장 감지 데몬 수행

위의 [그림4]는 Active 와 Standby 측의 고장 감지를 위한 데몬 수행 모습이다. standby 고장 감지 데몬은 관리자가 초기 설정한 WTL(wait time level - active 데몬의 응답 시간) 값을 가지고 active 데몬에 HCMS(hearbeat check message) 메시지를 보내며, Active 모듈은 HCMA 메시지로 응답을 한다. 만일 standby 모듈이 WTL * @ 값 이내에 HCMA 메시지를 받지 않는다면 Active 데몬이 고장이라라 인식하여 이중화 재구성 절차로 들어가게 된다.[6]

두 번째, 본 논문에서의 이중화 재구성은 2절 이중화 배경에서 제시한 방법을 기본으로 사용하며, 추가로 관리자가 롤 백 포인트를 지정하여 처음부터 다시 서비스를 시작하지 않고 지정해둔 포인트 지점으로 돌아가 기존의 서비스를 제공한다. 아래의 [그림5]는 [그림2]를 기본으로 한 고장시 이중화 재구성 데몬의 수

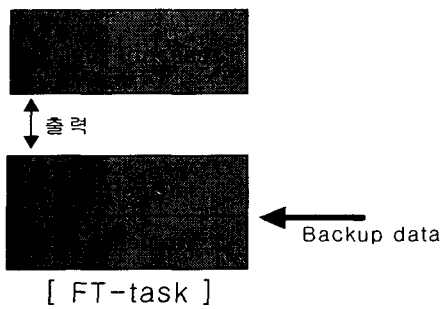
행 모습이다.



[그림5] Fault Recovery 데몬 수행

록 백 포인트를 지정함으로써 신속한 서비스를 제공할 수 있다는 것이 특징이다.

마지막으로, 리눅스에서 데이터를 이중화하기 위해서는 일반 리눅스 테스크 대신 이중화를 위한 전용 고장 허용 태스크를 만들어야한다. 일반 리눅스 테스크는 수개의 프로세스들이 동작하고 있기 때문에 이 모든 데이터를 저장하기에는 정보의 양이 많고 구현이 어렵기 때문에 리눅스 이중화를 위한 태스크를 따로 만들어 기존의 리눅스 테스크와 상호 연동을 해야 한다. 아래의 [그림6]는 리눅스 프로세스와 고장 허용 테스크와의 상호 연동을 보여준다. [7]



[그림6] Fault-tolerance task 구조

이중화 데이터 처리는 FT-task가 맞고 출력은 기존의 태스크가 맡아 처리한다. 또한 FT-task의 구성은 이중화 데이터 처리에 필요한 상태 레지스터와 dirty memory(최소 memory)로 이루어지며 관리자가 지정해 둔 롤 백 포인트 마다 처리 데이터가 FT-task에 의해 백업되게 된다.

Ⅳ. 결론 및 향후 연구과제

이번 논문은 리눅스 시스템에서의 이중화 시스템 구축을 위한 연구이며, 본 논문은 하드웨어중의 standby sparing 구조를 기본으로 한다. 또한 이중화를 위한 기본 고려 사항인 고장 감지와 이중화 재구성, 이중화 데이터 처리를 분석하여 이를 토대로 리눅스 시스템에서의 이중화를 위한 리눅스 이중화 모듈을 설계한다. 이중화 모듈 설계시 가장 중요한 것은 안정적으로 데이터를 처리하는 것으로 FT 테스크를 따로 두어 전문적으로 이중화 데이터를 처리한다. 결론적으로 리눅스 이중화 모듈을 자유롭게 리눅스 커널에 올리고 내림으로써 차후 리눅스 이중화 시스템의 변경에도 유연성 있게 대처 할 수 있다.

향후 연구 과제로는 본 연구에서 제시하고 있는 리눅스 이중화 모듈을 실제 리눅스 환경에서 구현하여 시스템의 안정성 검증이 필요하며, 이중화시 데이터 처리에 대해 좀 더 효율 적인 처리 방안에 대한 연구가 필요 하다.

참고문헌

[1] 세림정보기술(주) (www.selim.co.kr/duplex/intro.htm)

[2] Barry W.Johnson, "Design and Analysis of Fault-Tolerant Digital Systems", Addison-Wesley Publishing Company, 1989.

[3] 시사컴퓨터,(www.sisait.co.kr/199909/choice/150. htm)

[4] 고광호 외, "ATM 교환기 운용 웍스테이션 이중화 제어 및 관리 방법", 한국전자통신연구원 ATM운용보전팀.

[5] 이용석 외, "PCL 고장허용에 대한 이중화 시스템 연구", 한국전자공학회 v14,No.1,pp.47~52 January 2000.

[6] 이동호, "고장 감내형 실시간 제어 소프트웨어 설계", 전자기술연구지, v15-1,1994.

[7] Gannon, T.G외 "An optimal approach to fault tolerant software systems design", IEEE Transactions an Software Engineering, V.SE-4, No.5, September, 1978