이중 채널 이더넷을 이용한 분산 결함 허용 시스템

최보곤°, 김진용, 함명호, 신현식
서울대학교 컴퓨터공학부
{bgchoi°, sugar, mhham}@cselab.snu.ac.kr, shinhs@snu.ac.kr

Distributed Fault-Tolerant System using Dual Channel Ethernet

Bogon Choi°, Jinyong Kim, Myoungho Ham, Heonshik Shin
School of Computer Science and Engineering, Seoul National University

요 약

고가용성 및 고신뢰성의 분산 결함 허용 시스템의 설계와 구현에 대해서 다룬다. 이 시스템은 관리자 노드와 작업 노드 풀로 노드들을 구성하고, 각각의 노드들은 결함 허용 네트웍을 통해 통신을 하게 된다. 이 결함 허용 네트웍은 두 개의 네트웍이 중복되게 구성되어 한 네트웍의 결함 시에도 정상적인 데이터 교환을 보장한다. 여기서 중복된 네트웍을 위한 결함 검출 복구 기법이 필요하고 이들 관리자 노드와 작업 노드들의 관리를 위해 결함 허용 미들웨어가 포함된다. 미들웨어의 기능에 적응형 결함 허용 기법을 도입하여 실행 시간에 결함 허용 모드를 선택할 수 있게 하고, 결과적으로 보다 높은 가용성과 신뢰성의 결함 허용 시스템을 구성하였다.

1. 서 론

네트웍 기술의 발전으로 인해 데이터는 여러 장소에 분산되어질 수 있게 되었고, 빠른 네트웍과 시스템 성능의 향상으로 이들의 접근에 대한 투명성을 제공하게 되었다. 사용자가 네트웍에 의존하는 시스템의 수들도 증가하게 되었는데, 범용 시스템은 물론이고 시간 제약이 심한 실시간 시스템인 항공 관제 시스템, 비행 제어 시스템, 군사적인 용도의 무기체제 시스템에까지 폭 넓게 네트웍 기술은 이용되고 있다. 이런 다양한 시스템에서 네트웍은 명령을 전달하거나 측정된 데이터를 주기적으로 교환하는데 없어서는 안될 중요한 시스템 부분이라고 할 수 있다. 하지만, 큰 규모의 네트웍일수록, 고속의 네트웍일수록 결함의 발생은 피할 수 없다[3]. 이런 네트웍을 통해 노드들은 많은 양의 명령들을 처리해야 하는데, 특히 시간 제약이 심하거나 데이터의 내용이 중요시되는 시스템들은 처리된 데이터를 높은 신뢰성을 가지고 목적지에 다시 전달해 주어야만 한다. 이런 요구 사항을 만족시키기 위해서는 결함 허용 기법이 적용되어져야만 한다. 여러 검증된 상용의 결함 허용 시스템들이 주로 중복 기법을 통해 만족스러운 신뢰성과 가용성을 보장하고 있다[5]. 시간의 제약이 있는 시스템이라면 H/W나 S/W의 중복이 필수적이고, 네트웍 인터페이스 카드(이하 NIC)의 중복, 명령을 처리할 노드와 해당 프로세스의 중복이 일반적이다.

물리적 계층의 중복을 통한 결함 허용 네트웍에 대한 연구는 과거 많이 이루어졌다. MARS 자동 시스템은 이더넷의 중복을 통해 LAN을 구성하였고, AAS 항공 교통 제어 시스템은 다수개의 토큰 링을 사용하여 신뢰성을 보장하였다[4]. 이들 연구는 특정한 목적으로 제작된 하드웨어를 사용했기 때문에 개발 기간이 길고 장비의 가격이 높아지는 단점을 지녔는데, 근래의 연구에서는 기존의 상용 제품(Commercial-Off-The-Shelf, COTS)을 이용한 연구가 진행되어 LAN에서의 결함 허용을 지원하고 있다[1]. COTS NIC, 허브, 스위치의 사용은 과거 연구의 단점들을 해결하는 방법을 제공한다. OFTE와 FTE는 COTS 이더넷 NIC를 중복되게 사용하고 네트웍 계층2의 위치에 미들웨어를 구성하여 LAN상에서 결함 허용 네트웍을 구성했다. 네트웍의 노드 간에 지속적인 데이터 교환을 위해서는 해당 서비스를 계속 유지시키는 것 또한 중요한데 이런 서버에 대한 고가용성 연구는 다음과 같다. Linux-HA(High Availability)연구의 하트비트(Heartbeat)는 결함 검출 기법을 사용해 웹서비스 등에 대해서 결함 시, 노드 간 전환(Take-over) 기능을 제공한다[6]. 리눅스 가상 서버(Linux Virtual Server, LVS)와 UltraMonkey 프로젝트에서는 한 개 이상의 리눅스 가상 서버와 다수 개의 리얼 서버(Real Server)들로 구성이 된다. 가상 서버는 리얼 서버 풀(Real Server Pool)을 관리하며, 서비스에 대한 요청이 왔을 경우, 이를 선택된 리얼 서버에 전달(forwarding)하는 작업을 수행한다[그림1].

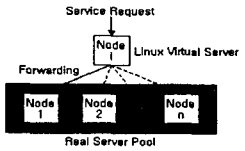

그림 1. 리눅스 가상서버

기존의 연구들에서 고가용성의 클러스터링 연구가 많이 진행되었지만, 데이터가 교환되는 네트웍 결함에 대한 중요성은 인식하지 못했다. 그 이유는 임무의 중요도가 상대적으로 가벼운 환경을 가정했기 때문인데 이 경우 네트웍에 결함이 발생하면 전체 시스템의 서비스가 중단될 수밖에 없는 상황이 발생한다. 하지만 본 연구는 군사적 목적과 같은 보다 엄격한 임무 상황을 고려하고 있고, 결함 허용 네트웍을 서버 클러스터링에 적용시키고 있어 보다 높은 가용성과 신뢰성을 보장하게 된다.

본 연구에서는 높은 신뢰성과 가용성을 동시에 만족시키는 분산 결함 허용 시스템의 설계와 구현을 제안할 것이다. 구성

될 시스템의 각각의 노드들은 COTS NIC의 이중 채널로 구성되고, 관리자(Supervisor) 노드와 작업 노드 풀(Operator node pool)을 구성하여 노드 결함에 따른 서비스 장애 시에도 계속적인 서비스를 보장하게 한다. 이들 시스템의 운영을 위해 미들웨어를 설계하여 사용할 것이며, 관리자 노드와 작업 노드는 이 미들웨어를 통해 명령을 교환한다. 관리자 노드가 작업노드를 선택할 때, 적응형 결함 허용 기법(Adaptive Fault Tolerance)을 적용하여 기존의 라운드 로빈(Round Robin) 스케줄링에서 벗어나 보다 유연한 결함 허용 기법을 작업 노드에 적용할 수 있게 한다. 그 결과로 기존의 시스템에서 제공하지 못한 높은 가용성과 신뢰성을 제공하게 된다.

2. 결함 허용 시스템의 구조

2.1. 이중 채널의 결함 허용 네트워크 구성

긴박한 임무 상황(mission critical)에서 제어 명령이나 주기적인 데이터를 교환하기 위해서는 지역적인 제어 네트워을 구성하는 것이 필요하다[3]. 이 네트워은 결함 허용성을 지녀야 하는데 한 곳의 네트워크 결함은 네트워크 전체에 데이터 통신의 중단이라는 치명적인 결과로 나타날 수 있기 때문이다. 그러므로, 두 개 이상의 독립적인 네트워을 중복으로 구성하여 네트워에 결함이 발생한 상황에서도 나머지 네트워을 이용해 결함을 극복하는 방법이 중요하다[1]. 그림2는 본 연구에서 사용되는 네트워에 대한 간략한 정보를 제공한다. 내부 네트워은 액티브 네트워과 스탠바이 네트워으로 중복되게 구성하는데, 두 개의 COTS 허브, 노드들은 두 개의 COTS NIC으로 네트워에 연결된다. 발생할 수 있는 네트워 결함을 가정한다면 (1) 스위칭 허브 결함, (2) 네트워 링크 결함, (3) NIC 결함을 들 수 있다. 액티브 네트워에서 결함이 발생할 경우 스탠바이 네트워으로 전환하여 결함을 극복하며 높은 신뢰성을 제공한다.

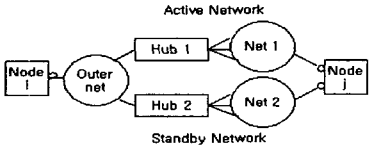

그림 2. 이중 채널의 결함 허용 네트워

2.2. 분산 결함 허용 시스템의 구성

높은 가용성의 시스템은 잦은 동작의 이상뿐만 아니라 얼마나 신속하게 복구되느냐에 의존적이다. 실제 서비스를 제공하는 노드에서 동작의 이상이 발견되더라도 여분의 노드들에서 서비스를 복구시킨다면 가용 상태를 계속 유지하게 된다. 두 개의 채널을 가진 노드들은 결함 허용 네트워을 통해 통신하고 기능별로 각 노드들은 관리자 노드와 작업 노드로 나눠진다. 관리자 노드에서는 작업 노드 풀을 관리하며 외부로부터 전달되는 요청을 선택된 작업 노드에게 전달하는 역할을 한다. 작업 노드 풀은 현재 서비스 가능한 노드들이며 같은 동작을 수행하는 프로세스들이 준비된다. 여러 버전의 소프트웨어를 중복시키는 것은 결함 허용 기법으로서, 결과적으로는 제공되는 서비스의 신뢰성과 가용성을 높인다. 실행 결과는 관리자 노드에게 반환된다. 작업 노드를 선택하는 방법에는 라운드 로빈 방식 등이 존재하지만, 이를 결함 허용 기법이 적용되게끔 확장하여 P/B(Primary/Backup), DRB(Distributed Recovery Block), NVP(N-Version Programming) 기법을 사용할 수 있게 한다. 그림 3은 결함 허용 네트워을 포함한 분산 결함 허용 시스템의 구성을 보여준다. 여기서 관리자 노드는 단일하게 구성되었으나 이 역시 중복되게 구성할 수 있다.

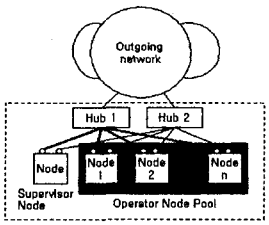

그림 3. 분산 결함 허용 시스템

3. 분산 환경의 결함 허용 미들웨어

3.1. 결함 허용 미들웨어 구조

미들웨어는 네트워 결함의 검출과 복구, 작업 노드들의 관리, 결함 허용 통신, 적응형 결함 허용 기법을 이용하여 분산 결함 허용 시스템을 운영한다. 미들웨어를 커널 내부에 두지 않고, 애플리케이션 계층에서 설계하여 기존의 실시간 운영체제와 범용 운영체제에 이식성을 높이게 설계되었다. 관리자 노드와 작업 노드들에서 동작하며 논리적인 구조는 다음의 그림4와 같다.

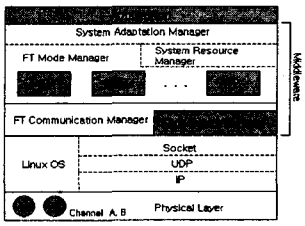

그림 4. 결함 허용 미들웨어의 구조

물리적 계층에는 두 개의 채널을 구성하여 각각의 채널에는 IP가 부여되고 채널을 구분하게된다.

- 결함 검출 관리자(Fault Detect Manager): 네트워 상의 결함과 작업 노드들의 상태를 하트비트 메시지를 통해 파악한다. 네트워 결함 시, NIC을 전환하는 방식으로 스탠바이 네트워을 활성화시키고, 노드 결함 시는 작업 노드 풀에서 해당 노드를 제외시켜 복구한다.
- 결함 허용 통신 관리자(FT Communication Manager): 시그널을 통해 결함 발생이 확인되더라도 스탠바이 채널을 통해 통신의 신뢰성을 유지시킨다.
- 시스템 자원 관리자(System Resource Manager): 관리자 노드와 작업 노드 풀에 대한 최신 정보를 관리하고, 시스템 내부 모니터 기능을 제공하여 이를 시스템 외부에 공급한다.
- 결함 허용 기법 관리자(FT Mode Manager): 작업 노드 풀에 적용할 결함 허용 모드들이 준비되어 있다. P/B, DRB, NVP 방식을 지원하고, 결함이 필요없는 응용을 위

해서 기존의 라운드 로빈 방식도 지원한다.

- 시스템 적응 관리자(System Adaptation Manager): 시스템 자원 관리자의 정보와 사용자 명령을 분석하여 결함 검출 기법 관리자의 기법 중 하나를 선택한다.

3.2. 결함 허용 미들웨어 구현

각각의 노드는 리눅스 커널 2.4.7 버전의 운영체제가 동작하는 환경이다. 결함 검출 관리자는 하트비트(Heartbeat) 프로토콜을 사용하여 네트웍과 노드들의 결함을 검출해 낸다. 관리자 노드에서 메시지를 보내며 작업 노드 풀의 노드들은 이 메시지에 응답하여야 하는데, 두 개의 채널에 동시에 메시지를 보내기 때문에, 하트비트 메시지의 개수는 $2n$개가 된다(노드 개수 n). 여기에 메시지 누락(Message ommission) 문제를 막기 위해 복수 개의 하트비트 메시지 α개를 중복해 보낸다[4]. 결함 검출 시간은 하트비트 간의 시간 간격에 의존하는데, 다음 하트비트 간의 시간 간격 τ와 메시지 지연시간 δ의 합이다.

메시지 개수 = $2n \times \alpha$

결함 검출 시간 = $\tau + \delta$

이 하트비트 프로토콜을 통해 관리자 노드는 분산 시스템에서 사용하는 AST(Active Station Table) 정보를 유지하게 된다. 그리고, 시스템에 결함 발생이 감지되면, 시그널을 발생시켜 시스템에 알리게 된다. 발생하는 시그널은 다음과 같다.

(a) CH_A_FAIL: 채널A 결함 발견
(b) CH_B_FAIL: 채널B 결함 발견
(c) CH_ALL_FAIL: 모든 채널의 결함
(d) NODE_FAILi: 특정 노드i의 결함

네트웍 결함의 경우 결함 복구 기법은, 작업 노드 풀에 해당 채널의 결함을 통보하여 네트웍 노드들의 NIC을 스탠바이 채널로 전환시킨다. 전체 노드들은 스탠바이 채널을 통해 메시지를 교환하게 된다. 노드 결함의 경우 AST에서 해당 작업 노드를 제외시켜 시스템 적응 관리자가 작업 노드를 선택할 수 없게 한다.

결함 허용 통신 관리자는 다음의 기본적인 통신 함수를 제공하며, 이들 함수들은 RTO(Receive Time-Out)을 이용한 재전송으로 신뢰성 있는 통신을 보장하게 된다. 그리고, 추가적인 CRC 에러 체크 기법을 도입하여 메시지 에러에도 대처한다.

ft_prim_send{}: 기본적인 신뢰성 전송
ft_prim_recv{}: 기본적인 신뢰성 수신
ft_send_recv{}: 재전송과 CRC를 통한 요청, 응답
ft_broad_send_recv{}: 작업 노드들에 브로드케스트

시스템 적응 관리자의 의사 결정에 대한 내용은 그림 5와 같다. 의사 결정에 필요한 정보로는 애플리케이션 종속적인 정보와 시스템 자원 관리자로부터 입력이 있다. 전자의 정보는 애플리케이션 디자인 시에 이미 주어지게 되고, 후자는 미들웨어의 상태 변화에 대한 것으로서 시스템 자원 관리자로부터 전달되게 된다. 이들 입력을 기반으로 결함 허용 기법 관리자에게 P/B, DRB, NVP 등의 선택된 기법을 통보한다. P/B의 경우, 기본 작업 노드를 AST에서 선택하여 명령을 전달하고, 워치독(Watchdog)이 지날 경우, 백업 작업 노드에게 수행을 스케줄링 한다. NVP의 경우, N개의 작업 노드에 동시에 프로세스 수행을 요청하고, 결과는 관리자 노드에서 취합하여 투표(Voting)하게 된다[5].

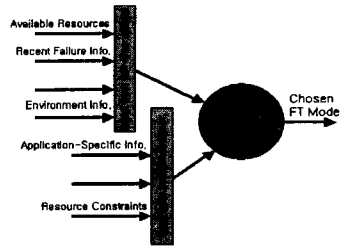

그림 5. Adaptation Decision Making

4. 결과 및 향후 연구

본 연구에서는 COTS 네트웍 제품의 중복을 이용하여 결함 허용 네트웍을 구성하고, 각각의 노드에 미들웨어를 구성하였다. 이들 노드의 작업은 관리자 노드에서 작업 노드 풀로 명령을 전달하고 데이터를 다시 전달받는 방식으로 동작한다. 이런 클러스터링을 구성한 방식은 이미 많은 연구에서 이미 이뤄졌었다. 하지만, 네트웍 상의 결함에 대한 고려의 부재와 노드를 선택하는 스케줄링 방식의 한계로 고신뢰성 및 고가용성이 요구되는 분산 환경에서는 적절치 못한 상황이었다. 이를 개선하고, 정적인 작업 노드 스케줄링 방식에서 벗어나 적응형 결함 허용 기법(Adaptive Fault Tolerance)을 적용하여 동작 환경에 따라 유연하게(flexible) 스케줄링을 적용시키는 방식으로 연구하였고 결과물을 얻었다. 앞으로의 연구는 두 개의 채널을 확장하여 n개의 채널에 대한 결함 검출 기법의 연구와 이를 적용한 분산 결함 허용 미들웨어의 지원에 있겠다.

참고 문헌

[1] Huang, J.; Song, S.; Li, L.; Kappler, P.; Freimark, R.; Gustin, J.; Kozlik, T., "An open solution to fault-tolerant Ethernet: design, prototyping, and evaluation", Performance, Computing and Communications Conference, 1999 IEEE International , 1999 Page(s): 461 -468

[2] Hecht, M.; Hecht, H.; Shokri, E., "Adaptive fault tolerance for spacecraft", Aerospace Conference Proceedings, 2000 IEEE , Volume: 5 , 2000 Page(s): 521 -533 vol.5

[3] Jae Min Lee; Wook Hyun Kwon; Young Shin Kim; Hong-Ju Moon, "Physical layer redundancy method for fault-tolerant networks", Factory Communication Systems, 2000. Proceedings. 2000 IEEE International Workshop on , 2000 Page(s): 157 -163

[4] Sape Mullender, "Distributed Systems, 2nd Ed.", Addison-Wesley, 1993

[5] Dhiraj K. Pradhan, "Fault-Tolerant Computer System Design", Prentice Hall PTR

[6] High Availability Linux Project, http://www.linux-ha.org