2000년 한국정보처리학회 추계 학술발표논문집 제7권 제2호

SPAX를 위한 결함허용 파일서버에 관한 연구

o
진 광윤*, 한 판암**
*삼척대학교 컴퓨터공학과
**경남대학교 컴퓨터공학과
e-mail : kyjin@samchok.ac.kr

A Study on Fault Tolerant File Server for SPAX

Kwang-Youn Jin*, Pan-Am Han**
*Dept of Computer Engineering, Sam-chok University
**Dept of Computer Engineering, Kyoung-nam University

요약

최근 SPAX와 중대형 컴퓨터 시스템에서 OLTP와 DSS와 같은 대용량 고속 자료 처리 서버로서의 자료 무결성 문제는 기술적으로 선결되어야 할 중요한 사용자 요구 사항 중의 하나이다. 자료 무결성에 대한 기술적 해결의 한 방안으로 응용 프로그램 수준의 결함허용 기법을 제공하기도 한다. 그러나 응용 프로그램 수준의 결함허용 기법은 자료를 저장한 매체의 관리 권한 밖에 존재함으로써 고장의 발생과 복구에 일정 제한을 가진다. 따라서 디스크와 같이 일반 파일 시스템을 구성하는 중요한 매체들에 저장된 자료의 무결성을 커널 수준에서 보장하고 응용 프로그램에 투명성을 보장 할 수 있는 연구가 향후 기술 동향의 한 축을 이루고 있다. 본 연구에서는 마이크로 커널을 기반으로 상위 서버로서 결함허용 파일서버의 설계에 관한 내용을 제안한다.

1. 서론

현재 커널 수준의 결함허용 시스템 및 결함허용 파일 시스템 설계에 대한 연구는 고도의 기술성에 대한 완전한 문제의 해결 (Solution)을 찾지 못하고 있으며, 또한 성능에 대한 Trade-off 등으로 상용화 단계로 접어들지 못하고 있다. 본 연구에서 제안하는 결함허용 파일 서버 설계를 위하여 먼저 마이크로 커널을 연구하며 상위메시지 통신포트를 갖는 운영체제 Server를 설계한다.

이 서버는 마이크로 커널의 특성(Features)을 사용 하여 다른 서버들과 메시지로 통신하는 결함 허용 파일 서버를 설계함에 있어 같은 Disk pool을 공유하는 프라이머리(Primary) 서버와 백업(Back-Up) 서버를 두어 프라이머리 서버의 결함은 즉시 백업 서버가 그 작업을 인수(Take-over)하므로서 결함 복구는 물론 자료의 무결성을 보장하는 기법들을 연구한다. 결함의 발생과 통보는 별도의 결함 관리 서버(Fault Management Server)를 두어야 할 것이며, 플랫폼 또한 단일 노드는 물론 다중 노드에서 운용이 가능하도록 설계한다.

Ⅱ. SPAX/FT의 결함 관리 기법

Scalable Parallel Architecture compute based on X-bar network(SPAX) 시스템의 결함허용 기술(SPAX/FT)은 Tandem 의 "Non-stop machine"과 비슷한 분산환경 하에서 process-pair 방식을 사용하는데, 다른 점은 Tandem 이 같은 프로세스를 동시에 수행하는 Hot-spare 방식을 사용한다면, SPAX/FT는 같은 프로세스 중 한 프로세스는 계산을 수행하는 동안 다른 한 프로세스는 오류가 발생할 때까지 기다리는 Hot-standby 기술을 사용한다. 그러나 모든 프로세스에 이 기술을 적용하는 것이 아니라, 시스템의 중요한 서버에만 적용시킨다. 그리고, 각 서버나 노드의 상태를 관리해주는 Fault Tolerant Manager(FTM)을 사용하여, 총괄 관리함으로서 시스템의 소프트웨어 및 하드웨어의 결함에 대한 신뢰성 향상에 그 목적을 두고 있다.

본 연구의 가장 핵심은 결함 처리자(fault handler)와 S&T(Switch and Take-over)이며, 연구의 목표는 한 노드의 오류(커널의 오류)로 인하여 시스템 전체가 정지하는 것을 방지하는데 있으며, 따라서 각 노드에 올라가 있는 SSU 서버들 중 시스템 전체적으로 중요한 일부 서버들을 이중화(process-pair)하여 적재한다.

2.1 오류 복구하는 과정

본 연구의 목표인 각 노드에 올라있는 SSU 서버들 중에서 시스템 전체적으로 중요한 일부 서버들을 프로세스를 이중화(process-pair)하여 적재한다.

Process-pair를 기반으로 해서 오류를 복구하는 과정은 아래와 같다.

(1) 먼저 오류감지(Error Detection)을 수행하여 마이크로 커널이나 각 서버들에 오류가 발생했는지를 점검한다. 만일 오류가 정정이 가능한 것이라면 결함 처리자(Fault Handler)로 하여금 오류를 정정하게 한다. 정정이 불가능하다면 오류가 다른 부분으로 전파되지 않도록 각 결함의 특성에 따라서 오류를 차단시킨다.

(2) 결함 정정 루틴으로 하여금 결함에 대한 정정을 시도한다. 만약 성공하면 발생한 영역이 정정이 되며 서버는 다시 정상 동작을 수행하게 될 것이다.

(3) 만일 오류에 대한 정정이 실패했다면, 오류를 고립시키기 위하여 오류를 차단시킨다.

(4) 만일 해당하는 서버가 process-pair로 구성되어 있다면 Backup 으로 S&T를 실시한다. 그렇지 않으면, 단지 서버를 다시 시작(server restart)한다.

(5) 오류의 발생 때문에 정지되었던 마지막 오퍼레이션의 수행을 계속한다.

각각의 서버 내부와 마이크로 커널 내부에서도 오류를 감지할 수 있는 코드와 발생한 오류에 대하여 처리를 해줄 수 있는 코드가 필요한데 이것은 진단 커널에서 언급할 것이다. FM 서버내에도 서버자체의 오류를 감지하며 처리할 수 있는 코드를 가지는데 오류에 대한 복구가 실패하게 되면, 해당되는 서버가 종료되게 된다. 그러나, FM 서버는 노드간에 process-pair로 구성되어 있기 때문에 하나의 FM 서버가 종료되더라도 그 기능을 다른 노드에 있는 백업 파일 관리자(FTFM) 서버가 맡아서 수행할 수 있기 때문에, 파일 시스템에 대한 사용자의 입장은 변함이 없다.

2.2 process-pair 기반의 결함복구 구성 요소

이 process-pair 기반의 결함복구를 위한 구성요소는 크게 세 부분으로 나누면 다음과 같다.

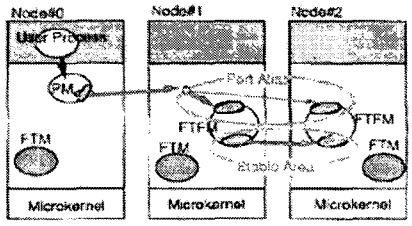
1) 마이크로 커널 부분
2) 서버 부분
3) 1) 2)를 총괄 관리해주는 결함 관리자(FTM).

1. 마이크로 커널의 결함 관리

마이크로 커널에서는 자체의 오류를 감지하는 결함 감지자, 오류에 대하여 정정을 수행할 수 있는 결함 처리자, 그리고 결함 VIEW 관리자(FTM) 와의 연계를 수행하는 마이크로 커널 결함 처리 배정자가 있으며, 마지막으로 시스템 전체의 생존해 있는 노드의 상태를 관리하는 "SITE VIEW"관리자가 있다.

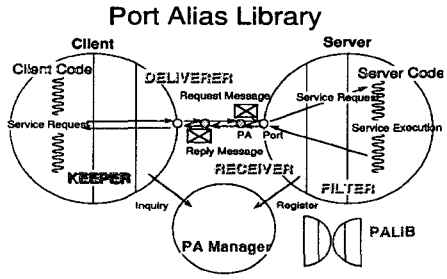
2. 서버의 결함 관리

각 서버에는 서버 자체내에서 발생한 오류에 대하여 감지할 수 있는 서버 결함 감지자, 발생한 오류에 대하여 정정을 수행할 수 있는 서버 결함 처리자, 그리고 이들을 결함 관리자와 연계시켜주는 서버 결함 처리 배정자로 구성되어 있으며, 결함 관리자에게 제공해 주는 안정 영역(Stable Area)과 "FT port"에 대하여 접근할 수 있는 라이브러리들로 [그림 1]과 같이 구성된다. [그림 1]은 3개의 노드에서 수행중인 몇 개의 중요 서버들에 대한 결함 관리를 위한 기법을 도시하고 있다. 사용자 프로세스가 수행중인 노드 #0를 클라이언트 노드이며, 결함허용 관점에서 Master 노드라 한다.



[그림 1] Server Layout

노드 #1과 노드 #2는 Slave 노드로서 각각 Primary 노드와 Backup 노드가 된다. [그림 1]에서는 노드 #1이 Primary로 동작하고, 만일 Primary 노드의 파일 관리자 (FTFM)에서 결함이 발생하면 Port Alias에 의하여 Back-up 노드인 노드 #2의 FTFM에게 자동적으로 메시지가 전달된다. [그림 2]는 Port Alias Library의 Components 들에 대해 도시하였다.



[그림 2] Port Alias Library

PALIB(Port Alias LIBrary) 는 Entity에 투명하게 request를 전달하는 기능을 하고 있다.

그 외에 결함허용에 관련된 기능으로 Node나 Manager의 오류시 메시지를 연계되는 다른 Entity에 투명하게 전달하고 중복되는 메시지를 처리하는 기능을 제공하는 역할을 한다.

(1) PALIB는 다음과 같은 기능적 요구 사항들을 포함한다.

1) 앤터티의 포트로 메시지를 자동 라우팅 시키는 기능.
2) 현재 앤터티가 이상이 있을 때 재수행을 위해 요구 메시지를 자동 재전송하는 기능
3) 중복 메시지 처리 기능.
4) 새로운 클라이언트가 생성되었을 때 메시지를 재 전송하는 기능

3. 결함 관리자(FTM)

마이크로 커널의 결함 관리, 서버의 결함 관리등을 총괄해서 관리해주는 관리자(FTM)의 모듈은 다음과 같은 역할을 수행하게 된다.

(1) SSU Server view

이것은 각 서버들이 어느 site에서 돌고 있나를 알 고 있어야 하므로. 각 사이트의 서버들에 변화가 생겼을 때 그 변화도 즉시 알아차릴 수 있게 한다.

(2) FT Port Management

Primary port 와 Backup port를 위한 alias가 기록되어 있는 mapping table 을 관리한다.

(3) Recovery Management

process-pair에 대해 복구 과정을 관리하며, process-pair를 관리하기 위한 actor를 생성하거나 초기화시킨다. 그리고, FT 기능이 없는 클라이언트들의 오류를 보고한다.

(4) Stable Area management

name과 port가 짝으로 구성되어 있는 Stable Area의 name space를 관리한다.

Ⅲ. 결함허용 파일 시스템(FTFM)

Microkernel Based Single System Image Unix (MISIX)에서의 결함허용의 초점이라 할 수 있는 파일 시스템에 결함허용을 구현한 결함허용 파일 시스템 관리자에 대하여 기술하면 다음과 같은 내용이 포함된다.

3.1 결함허용 파일 시스템 관리자 기본 구조

결함허용 파일 시스템 관리자는 안전영역(SA) 와 포트가명(PA)을 기본 바탕으로하여 서로다른 노드상에 primary, backup의 2개의 액터로 이중화 프로세스 방식으로 구현되며, 시스템이 일반적인 상태일 때에는, primary 액터가 서비스를 제공해 주게 되며, backup actor는 아무 일도 하지 않게 된다.

1. 안전영역(Stable Area)

이 안전영역(SA)은 primary 액터와 backup 액터 사이에 필요한 데이터를 전송하는데 사용되는 저장 장소이다. 만일 Primary 액터에 서비스중에 오류가 발생하였을 경우에, backup 액터에서 안전 영역에서 필요한 정보를 가져가 정상적인 서비스를 계속 수행할 수 있게 해준다.

2. 포트가명(Port Alias)

포트가명(PA)는 primary 서버에 오류가 발생하였을 경우에 ipcCall 의 목적지를 primary 서버의 포트로부터 backup 서버의 포트로 자동적 으로 변경해주는 기술이다. 따라서, primary 서버의 오류를 클라이언트는 볼 수가 없고, 사용되는 인터 페이스는 일반적인 포트와 동일하다.

3.2 오류에 대한 사용자의 관점

결함허용 파일 시스템은 대부분의 오류에 대하여 사용자 프로세스에서는 보이지 않지만, 아래의 오류에 대하여는 사용자에 보여주게 된다.

① 디스크상의 파일 시스템의 inconsistency

파일 시스템에 inconsistency 가 발생하게 되면, 파일 시스템 자체를 사용할 수 가 없게 되므로, 복구 프로세스는 의미가 없어지게 되며, 사용자는 해당되는 파일 시스템을 unmount 시키고 파일시스템 점검을 수행하여야 한다.

② I/O 오류

Disk 상의 I/O 오류는 하드웨어나, 디스크 드라이버에서 처리되고 숨겨질 수 있다. 그래서, 결함허용 파일 시스템 관리자(FTFM) 에서는 이러한 오류에 대하여 처리를 해주지않으며, 이러한 종류의 오류는 사용자에게 보여 주게 될 것이다.

③ 사용자 프로세스가 돌고 있는 노드에 crash 되었을 때의 사용자 정보의 유실

만일 site crash 가 발생하게 되며, 그 site 에서 처리되고 있는 사용자 프로세스의 데이터가 단지 해당 site 의 local cache 에만 존재한다면, 그 데이터는 유실 되게 된다.

3.3 결함허용 파일 시스템과 관리자 내부 구조

MISIX의 결함허용 파일 시스템에서는 sfs(ufs) 파일 시스템이 결함허용을 위한 프로세스 이중화 방식으로 구현된다.

(1) 프로세스 이중화

MISIX 결함허용 기법에서는 sfs(ufs) 파일 시스템의 결함허용 구현을 위하여 프로세스 이중화 방식을 사용한다. 따라서, primary FM과 backup FM 방식을 사용한 두개의 파일 관리자(FM)가 서로 다른 노드에 존재하게 되며, 안전영역을 사용하여 복구에 필요한 데이터를 저장하며, 포트 가명(PA)를 사용하여 클라이언트로 하여금, primary 로부터 backup 으로 takeover 가 되는 상태가 보이지 않게 해 준다.

(2) 복구 프로시져

복구 프로시져는 다음과 같이 3개의 프로시져로 구성되어 있다.

1) intra 서버 데이터에 대한 복구

Intra 서버 데이터의 복구의 프로시져들이 단일 쓰레드에 의하여 아래의 순서로 수행되게 된다.

① backup 서버 초기화

② 파일시스템과 관련되지 않는 vfs 데이터의 재구성

③ 파일 시스템 관련 데이터의 복구

④ 각각의 파일시스템에 대한 cleanup

⑤ 파일 시스템과 관련된 vfs 데이터의 재구성

2) Inter 서버 데이터에 대한 복구

Intra 서버 데이터에 대한 복구가 끝난 뒤, backup 파일 관리자는 다른 서버들과 관련된 데이터의 consistency 를 복구하기를 시도한다.

3) 서비스의 재시작

서비스는 다음과 같은 순서로 다시 시작된다.

① FTM 에게 backup FM 이 서비스를 재 시작했다고 알린다.

② fsck 가 필요한 파일 시스템에 fsck 를 수행한다.

③ PA(Port Alias)에게 처리자(handler)를 연결시킨다.

(3) 서버 *Consistency* 복구(SCR)

SCR 은 다음과 같은 순서로 수행된다. Backup 파일 관리자는 non-FT 서버들이 가지고 있는 자원들이 여전히 유효한지에 대하여 점검한다.

(4) 프로세스 관리자(PM)

PM이 여전히 살아있는지 점검하여서, vnode 의 reference count를 정정한다. 프로세스 id가 여전히 유용한지 점검하여서 filock를 정리한다.

(5) 경로명 관리자(PNM)

node reference count를 정정하거나, pathname cache 를 flush 시킨다.

Ⅳ. 연구 결과의 활용 방안

본 연구의 결과로 고속병렬 컴퓨터(SPAX)의 결함허용에서 신뢰도 높은 복구를 해 주는 파일 서버의 설계에 관한 내용의 제안으로 결함허용 파일 시스템의 신뢰성을 높임으로서 인터넷 및 고속병렬 시스템과 실 시간으로 운용되어지는 각종 분야의 고속화 시스템의 테스트 및 운용에 활용하게 되며, 또한 지속적인 연구 지원으로 신뢰도 높은 결함허용 시스템의 시제품 제작이 가능 할 것이다. 향우 정보산업 전반에서 수요가 예측되는 결함허용 시스템 연구 및 개발의 기반 기술로 활용되리라 기대한다.

참고 문헌

1. Brain Randell,"System Structure for Software Fault Tolerance", IEEE Trans. on Software Eegineering, VOL, SE-1, NO. 2, 1975.
2. Herbert Hecht, "Fault-Tolerant Software", IEEE Trans. on Software Reliability, Vol. R-28, NO. 3,1979.
3. VERITAS,"Veritas Reliant Release Notes Release 1.0", Dec. 1996.
4. Mark Roddy, "Amadeus Fault Management", Design Walkthrough, June. 1995..
5. yuji IMAI, masahiro KOMURA, mitsuhiro KISHIMOTO, "Port Alias Basic Design Version 2.1", Sept. 14, 1995.
6. 김 문회,"결함허용 시스템의 고려 사항 및 동향",KISS Review,Vol. 11, NO. 3, 6. 1993. VOL. 11, NO. 3, 6. 1993, pp. 7 - 16.
7. 이철훈, 유승훈," 결함 허용 및 결합 멀티프로세스 ", KISS Review, VOL. 11, NO. 3, 6. 1993.