컴퓨터 시스템의 신뢰성 확보를 위한 고장 감내 시스템 연구

김대원, 박호림, 오병균
목포대학교 멀티미디어공학과

Research about Fault Tolerant system for ensuring the Reliability of computer system

Dae-Won Kim, Ho-Rim Park, Byeong-kyun Oh
Dept. of Multimedia, Mokpo National University

요 약

현대 사회는 인터넷과 네트워크의 발전과 더불어 많은 성장을 거듭해 왔고, 인터넷을 이용한 많은 서비스들은 계속해서 생기고 있다. 이러한 사회적인 배경에서 인터넷을 이용한 서비스에 대한 의존도도 또한 급격하게 증가하고 있다. 서비스를 언제, 어디서나 접근이 가능하게 함으로써 사용자들이 편리한 서비스를 제공받을 수 있게 되었지만, 이러한 서비스들이 안정적으로 제공되어야 한다. 고로 본 논문에서는 컴퓨터 시스템에 대한 신뢰성을 확보하기 위해 사용되는 기법 중 소프트웨어적인 고장 감내 시스템에 대해 알아보고 그에 따른 기법들을 조사하여 단일 시스템에서 구현해 볼 수 있는 방법 중 감시 프로세스를 이용하여 서비스를 안정적으로 서비스를 할 수 있는지에 대해 알아보았다.

1. 서론

컴퓨터는 초기에는 수치적 계산에 이용되었으나 컴퓨터의 성능이 향상되고 가격이 하락됨에 따라 다양한 분야로 응용이 확산되고 있다. 컴퓨터의 응용 분야는 크게 세 가지로 분류된다. 이들은 과학 기술적 계산 응용, 사무 자동화 응용 그리고 정밀 기계의 제어 응용 등이다.[1] 이러한 응용 분야의 증가와 함께 현대 사회는 인터넷과 네트워크의 발전과 더불어 많은 성장을 거듭해 왔고, 인터넷을 이용한 많은 서비스들은 계속해서 생기고 있다. 이러한 사회적인 배경으로 인터넷을 이용한 서비스에 대한 의존도도 또한 급격하게 증가하고 있다. 이와 같은 서비스가 언제, 어디서나 접근이 가능하게 함으로써 사용자들은 편리한 서비스를 제공받을 수 있게 되었지만, 이러한 서비스들이 안정적으로 제공되지 않았을 경우에는 편리함을 느꼈던 것 이상의 대가를 지불해야 됨은 누구나 알 것이다. 초기에 인터넷은 단순히 정보 교류의 방편정도였지만 현재는 그 영역을 넓혀 실시간 전화서비스 제공, 실시간 화상회의, 전자상거래와 같이 생활 전반에 걸쳐 다양한 서비스를 제공하기에 이르렀고, 전 세계의 서비스 제공자들은 금융, 의료, VOD와 같은 다양한 서비스를 하루 24시간, 1년 365일 동안 제공하기 위한 연구를 하고 있다.[2] 이러한 서비스는 사용자가 요구하는 정보를 어떻게 지속적으로 정확하게 제공하는가가 중요하다. 서비스에 대해 위협을 줄 수 있는 행위나 현상은 다음과 같은 세 가지로 나누어 볼 수 있다. 외부로부터의 공격, 시스템이 의존하고 있는 요소의 결함에 의한 고장, 마지막으로 자연 재해와 같은 사고이다. 자연적인 재해로 인한 사고 이외에는 사용자들에게 지속적인 서비스를 제공해야 한다. 자체 결함 중 하드웨어적인 결함은 기술의 발달로 인해 줄어드는 반면 소프트웨어적인 결함은 증가하고 있다. 고로 본 논문에서는 고장 감내 기법을 이용하여 컴퓨터 시스템에 대한 신뢰성을 확보하는 구조를 제안하였다. 소프트웨어적인 결함을 해결할 수 있는 고장에 대해서 분석해 보고 이에 대한 해결 기법들에 대해 알아보았다.

본 논문은 다음과 같이 구성이 되어 있다.

2장에서는 고장 감내 시스템에 대한 개요와 소프트웨어 고장 감내 시스템에 대해서 알아보고, 3장에서는 소프트웨어적인 결함을 해결할 수 있는 고장 감내 구조에

대해서 알아보도록 하고 4장에서는 결론을 내렸다.

2. 이론적 배경

2.1 고장 감내 시스템 개요

고장 감내 시스템이란 시스템 안에서 발생된 고장(하드웨어나 소프트웨어로 인한 고장)을 감지하고 그에 대해 대처함으로써 제공해야 하는 서비스들을 안정적으로 제공하는 시스템을 말한다.

서비스를 안정적으로 제공하기 위해서는 '서비스를 얼마나 정확하게 제공하느냐' 와 '얼마나 지속적으로 제공하느냐'의 두 가지 문제로 볼 수 있다. 이는 신뢰성과 가용성의 두 가지로 바꿔 말할 수 있다.

신뢰성이란 시스템이 서비스를 제공하는 중에 고장이 발생한다 할지라도 시스템 사용자가 요구하는 서비스를 정확하게 제공하는 결과를 얻을 수 있는 것을 말한다.

가용성이란 시스템이 제공하는 서비스의 연속성의 보장과 관련되어 있는데 고장으로 인해 정체된 시간으로 시스템의 가용성을 평가하게 된다.

시스템의 신뢰성과 가용성은 정보를 제공하는 시스템에 대해서는 중요한데, 이들에 대해 문제를 발생시키는 요인을 총칭하여 고장(fault)이라 부른다.

시스템의 신뢰성과 가용성을 높이기 위한 방법으로는 두 가지로 나누어 볼 수 있는데, 그것은 고장 회피(fault avoidance)와 고장 감내(fault tolerance)로 나누어진다.

고장 회피는 시스템의 고장을 사전에 방지(prevention)하는 것으로 하드웨어나 소프트웨어적으로 신뢰성이 높은 부품이나 프로그램으로 시스템을 구성하는 것을 원칙으로 하는데 시스템을 구성하기 전에 충분한 사전 테스트로 신뢰성을 높이고자 하는 기법이다.

고장 감내 기법이란 시스템의 고장은 발생할 수 있다는 전제로 이에 따른 대비를 하여 시스템의 정상적인 동작을 하도록 하는 기법이다. 이에 따라 고장의 유무를 알아낼 수 있는 고장의 검출(fault detection)이나 이에 따른 부가적인 하드웨어나 소프트웨어(redundancy)를 사용해야만 하게 된다.

고장 감내 기법은 접근 방법에 따라 하드웨어 측면에서의 고장 감내 기법과 소프트웨어 측면에서의 고장 감내 기법으로 나누어지게 된다.

하드웨어 고장 감내 기법은 과거부터 많이 연구되고 사용되어 왔다. 이 기법은 시스템이 고장이 발생시, 고장에 대한 대처를 하드웨어 모듈을 이용하여 해결하고자 하는 방법이다. 이러한 시스템을 구현하기 위한 방법으로는 동작하는 시스템이나 모듈과 동일한 구조의 부가적인 하드웨어 시스템 또는 모듈을 필요로 한다. 하나 이상의 하드웨어를 부가적으로 필요로 하고, 부가된 하드웨어는 예비로 원래의 하드웨어에 문제가 발생하면 그에 대한 대비로 동작되도록 구성되어 있다. 두 하드웨어가 동일한 동작을 수행하는 중 원래의 하드웨어와 예비 하드웨어의 작업 수행 결과를 비교함으로 고장 유무를 결정하고, 고장이면 예비 하드웨어로 동작을 하도록 하는 방법이다.

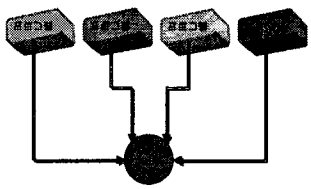
2.2 소프트웨어 고장 감내 시스템

소프트웨어 자체에 문제가 발생하는 경우는 여전히 시스템 오동작의 주 요인으로 자리잡고 있다.

이러한 소프트웨어 버그는 대부분이 프로그래머의 실수에 의해 이루어지기 마련인데, 크게는 디자인상의 버그(design bug)와, 코딩상의 버그(coding bug)로 나누어 볼 수 있다. 이러한 디자인이나 코딩 상의 버그는 이에 대한 수정이 이루어지기까지는 소프트웨어상에 존재하기 때문에, 항상 시스템의 고장 요인으로 존재하게 된다. 이러한 문제를 해결하기 위해서 디자인의 다양성(design diversity)을 이용한 N 변형 프로그래밍이나 복구 블록과 같은 방법 그리고 체크포인트 재시작에 대해서 알아본다.

2.2.1 N 변형 프로그래밍(N-version programming)

N 변형 프로그래밍(N-version programming)기법은 디자인의 다양성(design diversity)을 이용해서 시스템 디자인 단계에서 프로그래머의 오류를 최소화하는데 그 목적이 있다. 동작원리는 다음과 같다. 같은 기능을 수행하는 프로그램을 N 개 작성한 후 이 N 개의 프로그램을 동시에 동작시킨다. 작성된 N 개의 프로그래밍은 주어지는 입력에 대해 연산을 수행하고, 그 각각의 결과를 가지고 다수결의 원칙에 의해 정확한 결과가 무엇인지를 결정하는 방식이 바로 N 변형 프로그래밍 기법이다.



(그림 1) N 변형 프로그래밍

이 N 변형 프로그래밍은 다음과 같은 특징을 지니고 있다.

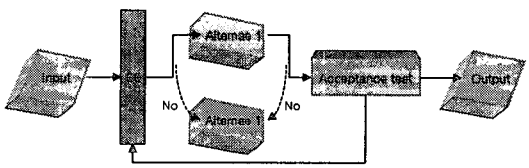
- N 변형 프로그래밍은 매우 강력한 기법이다.
- N 변형 프로그래밍을 사용해서 시스템을 구현하게 되면 시스템의 단가가 상승한다.
- N 변형 프로그래밍이 가지고 있는 또 다른 문제 중 하나가 평균 지능지수(average IQ) 문제이다.

2.2.2 복구 블록(Recovery Block)

복구 블록은 동일한 문제를 해결하기 위해서 여러 가지 대체 알고리즘을 사용하게 된다. 마치 스탠바이 스페어링 기법에서 예비 모듈(standby module)을 사용했던 방법과 비슷하다. 복구 블록의 동작원리는 다음과 같이 설명할 수 있다.

- 특정 프로그램 모듈을 수행하기 전에 앞서 그 모듈이 접근하는 데이터를 복사한다. 이 복사된 데이터를 recovery block 이라고 부른다.
- 특정 프로그램 모듈 A가 작업을 마치면 그 결과에 대하여 타당성 검사를 수행한다.
- 만약 타당성 검사에서 모듈 A의 결과를 받아들일 수 없다고 한다면, recovery block을 복사해서 프로그램 모듈 B에 넘겨주게 된다. 이 프로그램 모듈 B는 프로그램 모듈 A와 같은 기능을 수행하지만 다른 디자인으로 구성된 것이다.
- 특정 프로그램 모듈 B가 작업을 마치면 그 결과에 대하여 타당성 검사를 수행하게 되며 타당한 결과가 나올 때까지 모듈 C, 모듈 D 기타 등등에 계속 recovery block의 사본을 넘겨주게 된다.

이와 같은 방식으로 수행되는 복구 블록은 항상 적어도 한 개의 모듈은 정확한 결과를 출력하리라는 가정을 (그림 2)는 이러한 복구 블록을 간단한 블록으로 나타낸 것이다. 여기서 보이는 EE(Execution Environment)는 수행 환경을 나타낸 것으로 뒷 단의 타당성 검증 블록(Acceptance Test Block)으로부터 나온 신호를 참조하여 어떤 알고리즘 모듈을 선택할 것인지를 결정하는 역할을 수행한다.

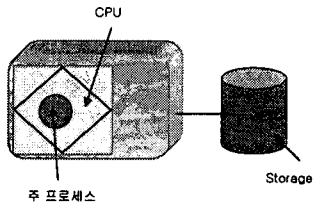


(그림 2) 복구 블록(Recovery Block) 모델

2.2.3 체크포인트 재시작 (Checkpoint and Restart)

체크포인트를 이용한 고장 감내 시스템은 고장 발생에 대비해서 흔히 사용되는 기법이다. 체크포인트 방식이란 수해중인 프로그램 중간중간에 체크포인트를 두어서 프로그램의 상태(status)를 신뢰할 만한 저장 장치에 저장한 후에 만약 시스템에 고장이 발생해서 시스템이 오동작을 수행하게 되었을 때, 프로그램은 다시 처음부터 동작을 수행하게 되는 것이 아니라, 가장 최근에 저장된 에러 없는 체크포인트 지점으로부터 작업을 재시작한다.

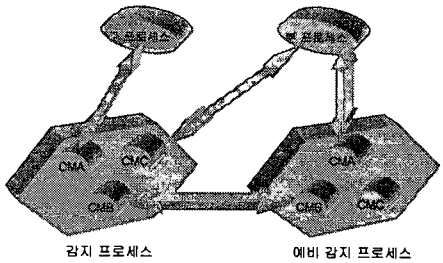
(그림 3) 은 이러한 체크포인트를 이용한 시스템을 간단한 구조로 나타낸 것이다. 가운데 마름모가 중앙처리장치(CPU)를 나타낸 것이고 그 안에 있는 원이 동작중인 프로세스를 나타낸 것이다. 이 프로세스는 옆에 보이는 저장 장치의 자신의 상태를 계속 저장하게 된다.



(그림 3) 체크포인트 방식을 이용한 블록도

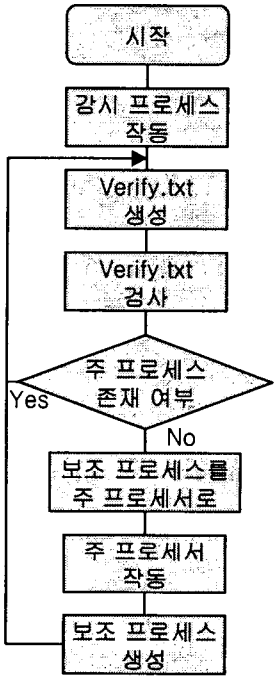
3. 고장 감내 단일 시스템 구조

감지 프로세스의 구조를 사용하였는데 감지 프로세스는 작업을 수행하는 주 프로세스 1:1로 신호를 주고받게 된다. 여기서 보면 주 프로세스와 예비 프로세스가 2 개의 모듈로 구성되어 있는 것을 알 수 있다. 이는 각 프로세스가 감지 프로세스와의 통신을 위한 모듈이 필요하기 때문에, 원래 작업 수행을 위한 모듈이외에 통신 모듈이 새로 첨가되었기 때문이다. 만약 주 프로세스에 오류가 발생해서, 감지 프로세스가 주 프로세스와의 통신에 실패하고, 프로세스가 만약 정상상태가 아니라고 판단을 내리면, 대기하고 있던 예비 프로세스가 그 동작을 대신 수행하도록 한다.



(그림 4) 감지 프로세스

- CMA(Communication Module A) : 고장 발생여부를 감지하고자 하는 주 프로세스와의 통신 모듈
- CMB(Communication Module B) : 예비 감지 프로세스와의 통신 모듈
- CMC(Communication Module C) : 주 프로세스에 고장 발생시 이를 대체할 예비 프로세스와의 통신 모듈



(그림 5) 작성한 고장 감내 프로그램의 작동 순서

4. 구현

본 연구는 서비스를 제공하는 역할을 서버에서 하므로 하드웨어는 Sun Microsystems 사의 UltraSPARC을 사용하였고 운영체제는 솔라리스를 2.6 버전을 사용하였다. 이러한 고장 감내를 위한 언어 작성은 C 프로그래밍을 이용하여 작성하였다. 우선 서비스를 정상적으로 하고 있다는 동작으로 아파치 웹 서비스를 위한 프로세스를 주 프로세스라 하고 실행시키도록 한다. 제대로 작동하는지 알아보기 위해 프로세스를 나타내는 명령인 ps 명령과 리다이렉션 기호를 이용해 verify.txt 로 저장을 한다. 저장된 파일의 내용을 작성된 고장 감내 프로그램을 이용해 프로세스가 작동중인지 검사를 한다. 검사 후 이상이 있으면 문제가 발생한 부분에 대해 작성된 프로그램에 따라 그 부분을 수정하여 다시 정상적인 시스템 작동 상태로 돌아온다.

본 논문에서는 강제로 프로세스를 종료시켜 작동중인 고장 감내 프로세스로 인해 서비스를 다시 실행함을 보였다.

4. 결론

인터넷과 네트웍의 빠른 변화로 인해 많은 서비스들이 등장하고 또 현재 사용되고 있다. 그만큼 서비스의 안정성은 이전과는 비교도 안될 정도로 크게 요구되고 있다. 본 논문에서는 그러한 문제를 가진 컴퓨터 시스템에 대해 신뢰성을 줄 수 있는 방안으로 감지 프로세스라는 개념을 이용하여 서비스 중인 주 프로세스에 대한 작동여부를 알아보고 문제가 발생하면 그에 적절한 조치를 위해 소프트웨어적인 프로그램으로 문제를 해결하는 방식으로 만들었다. 이후엔 최근 논의가 진행중인 침입 감내 시스템에 대해서 고장 감내 시스템과의 차이를 알아보고 침입 감내 시스템에서 고장 감내 시스템에서 적용했던 기법을 그대로 적용할 수 있는지에 대해 알아보도록 하겠다.

참고문헌

[1]E. Amoroso, "Instrusion Detection - An Introduction to Internet Surveillance, Correlation, Traps, Trace Back, and Response", Intrusion.Net Book, 1999.

[2]Amjad Umar, et. al., "Intrusion Tolerant Middleware", DARPA Information Survivability Conference & EXposition, 2001

[3]신동훈, "정보보증과 생존성 연구 동향", 한국정보보호진흥원, 2003

[4]양갑석, "소프트웨어 고장감내기법을 이용한 고가용성 시스템의 구현에 관한 연구", 전북대학교 정보통신공학과 학위논문, 2002

[5]Jim Gray, Andress Reuter, "Transaction Processing: concept and techniques"

[6]V. Stavridou, "Intrusion Tolerant Software Architectures," DARPA Information Survivability Conference & EXposition, 2001

[7]DARPA Information survivability program, http://www.darpa.mil/ito.