□특 집□

결함포용 병렬구조 시스템의 결함포용 지연시간 모델

포항공과대학 전산학과 김 종*

목 차

Ⅰ. 서 론

컴퓨터 시스템에 있어서 결함(fault)이 미치는 영향은 상당히 크다. 결함은 단지 시스템의 장애로 끝나는 것이 아니라, 시스템이 사용되던 환경에까지 영향을 미친다. 예를 들어 인명이나 재산의 피해를 가져올 수 있다. 이러한 결함으로 인한 피해를 최소화 시키기 위하여 결함포용 시스템들이 제안되고 설계되었다. 결함포용 시스템들이 이전에는 항공이나 통신 등 특히 고 신뢰성이 요구되는 분야에서만 사용되었으나, 요즈음에는 산업제어, 금융 등 그 응용분야를 넓혀가고 있다[1].

결함포용 방법으로는 하드웨어, 소프트웨어, 시간, 그리고 정보를 중복 사용하는 방법이 있다. 하드웨어를 중복하여 사용하는 결함포용 방법은 다시 3가지로 세분할 수 있다. 그들은 수동형(passive), 능동형(active), 그리고 복합형(hybrid)이다[2]. 수동형 방식은 결함에 의하여 발생되는 오류(error)의 파급을 막는 방법이다. 즉 오류가 발생하였을 경우 오류의 정정을 통하여 결함을 포용하는 방식이다. 이의 대표적인 예로 TMR (Triple Modular Rendundancy) 또는 NMR(N Modular Redundancy)이 있다. 능동형 방식은 결함 또는 오류의 검출로 결함이 있음을 확인하고, 위치 확인(fault location), 결함고립(fault isolation), 시스템 재구성(reconfiguration), 그리고 복구(recovery)를 통한 결함 부위의 제거와 시스템의 복구이다. 이러한 방법의 예로는 예비부품들을 준비하고 있다가 결함 발생시 교체시키는 예비시스템(spare system)이 있다. 복합형은 능동형과 수동형을 복합한 것으로 수동적 방법으로 오류를 정정함과 동시에 능동적 방법으로 결함이 발생한 부품(module)을 예비부품과 교체하는 방식이다. 이러한 결함포용 시스템의 전형

* 정회원

적인 예로는 FTMP(Fault Tolerant Multiprocessors)[3]을 들 수 있다.

병렬처리기(Parallel architectures) 분야의 발전은 결함포용 시스템 분야에 새로운 장을 제공하였다. 이는 동일한 자원이 중복적으로 존재하는 시스템에서 결함포용의 구현이 상대적으로 쉬워진 것 때문이다. 많은 병렬처리 시스템들이 능동형 또는 복합형 결함포용 방법을 채택하고 있다. 이러한 시스템들의 예로는 이미 상업적 시스템으로 널리 알려진 Tandem, Sequoia, Stratus, VAXft 등을 들 수 있다[4]. 이들 시스템들은 결함 발생시 동일한 성능을 유지한 채로 운영되거나(예, Stratus, VAXft), 성능의 하향조정 후 사용된다(예, Tandem, Sequoia). 후자의 경우를 gracefully degradable system이라고 한다.

결함포용 시스템의 결함포용 지연시간(fault-tolerance latency)은 결함발생으로부터 응용 program의 복구에 이르기까지 걸리는 시간을 말한다. 결함포용 지연시간의 중요성은 실시간 처리 시스템(real-time processing system)에서 찾아볼 수 있다. 결함포용 지연시간이 응용분야에서 요구하는 시간 제약 기준(dealine)에 부합하여야 결함포용 시스템을 사용할 수 있다. 즉, 예를 들어 최대 결함포용 지연시간이 t_1 millisecond이고 응용 program에서 실시간에 들어오는 입력이 t_2 millisecond 간격으로 들어온다고 할 때, $t_2<t_1$일 경우 입력손실로 인하여 시스템이 가지고 있는 정보가 부정확하여 질 수가 있다. 이는 곧 잘못된 제어(control)로 인하여 큰 피해를 가져올 수 있다. 따라서, 결함포용 지연시간의 분석은 결함포용 시스템의 응용분야를 결정하는 중요한 변수중의 하나이다.

본 고에서는 병렬처리 시스템의 결함포용 지연시간 분석모델에 대하여 논하고자 한다. 결함포용 지연시간은 세분하면 결함에 의한 오류의 발생으로부터 검출하는데 걸린 시간(detection latency), 결함부위를 고립하는데 걸린시간(isolation latency), 시스템 재구성에 필요한 시간(reconfiguration latency), 그리고 응용 program의 복구시간(recovery latency)들이다. 결함포용 지연시간 분석 중 부분적 분야에 있어서는 이미 많은 연구와 발표가 있었다. 예를 들어, 결함과 검출에 관한 model은 일반적 결함상태 전이에 관하여 이미 발표가 되었으며[5, 6], 오류 검출시간(detection latency)에 관한 연구도 일반적인 오류 검출방법을 이용하여 상태전이모델에 적용하는 방법에 대하여 발표되었다[5]. 결함부위의 고립에 관한 연구는 특별히 따로 연구되지 않고 있으며 결함검출의 일부분으로 연구되거나 결함부위를 포함하는 module을 고립시키는 것을 가정하고 있다. 시스템의 재구성에 관하여는 시스템이 가지는 환경, 재구성 구조의 선별시기, 재구성의 시점에 따른 연구가 발표되었다[7-9]. 복구에 관하여는 checkpointing이나 recovery block 등을 사용하여 결함이전의 상태의 내용을 복구하는 방법에 대한 연구[10-13]가 발표되었다.

본 연구분야는 앞으로 계속 연구, 조사중인 분야이고 현재 학계에서도 이에 대한 관심이 점차 증가하고 있다. 그 이유로는 병렬처리 시스템이 많은 중요한 응용분야에 사용되기 위하여 제안되고 있으며 그들의 공통적 요구사항은 결함포용 능력의 지원이기 때문이다.

II. 기존의 결함포용 능력 평가방법

2.1 방법

결함포용 능력의 평가기준은 응용분야의 목적에 따라 결정되어 왔다. 결함부품의 수리가 불가능한 환경하에서 신뢰도(reliability)을 평가하였으며 수리가 가능하면서 많은 서비스의 제공을 요구하는 분야에서는 가용도(availability)를 평가하였다. 이러한 분야의 예로서 전자는 항공, 우주선 등을 들 수 있으며 후자는 은행, 비행기 좌석예약 같은 시스템을 들 수 있다. 이 외의 평가항목으로써 시험도(testability), 유지도(maintainability), 성능 성취도(performability) 등이 존재한다. 위와 같은 시스템의 모든 평가 항목들을 통칭하여 의존도(dependability)라고 한다[13].

평가 결과를 나타내는 방법은 크게 세가지로 분류할 수가 있다[14]. 첫째, 시간, 또는 원하는 특징의 함수로서 나타내는 방식이다. 즉, 신뢰도

를 시간의 함수로서 표현하는 것이 이에 속한다. 둘째, 단일 변수로서 의존도를 나타낸다. 이 부류에 속하는 변수로서 평균 작동 시간(MTTF), 평균 수리 시간(MTTR), 장애간 평균 작동 시간(MTBF), Coverage 등을 들 수 있다. 세째, 상대비교함수 또는 변수로서 나타내는 방식이다. 이러한 예로는 신뢰도차(reliability difference), 신뢰도 이득(reliability gain) 등을 들 수 있다.

평가를 하는 방법은 네가지 방법으로 분류할 수 있다[15]. 첫째 방식은 신뢰도 블럭도(reliability block diagram)나 신뢰도 그래프(reliability graph)를 이용하는 방법이다. 이 방식은 일명 조합법(combinational method)라고 불리우며 모든 가능조합의 열거 후 합산으로 찾는 방식이다. 둘째 방식은 결함의존도(fault tree)을 이용하는 방식이다. 이는 모든 부품의 상호 의존도를 조사 분석후, 하나의 나무형식으로 의존도를 표시하는 방식이다. 이 또한 최종결과는 조합법으로 변환 후 얻어진다. 세째 방식은 Markov 모델 방식으로 부품의 고장발생율과 수리율을 일정한 비율로 발생하는 임의확률 프로세스(random process)라고 간주하여 시스템을 연속시간 유한상태의 정지 프로세스(stationary process)로 표현하는 방식이다. 네째 방식은 페트리 넷(petri nets)을 사용하여 시스템의 전이 상태를 표시하는 방식으로 최종결과는 simulation 또는 Markov 모델로 변환 후 얻어진다.

2.2 문제점

응용분야에서 요구하는 사항은 여러가지 있을 수 있다. 그 중 하나가 결함포용 능력일 것이다. 결함포용 능력에 대한 요구사항을 표시하는 방법에도 여러가지가 있다. 의존도는 결함포용 능력에 대한 하나의 방법으로 표시할 수 있을 것이다. 의존도에 의한 평가방법은 시스템을 장기적으로 운영할 때 가지는 시스템 속성의 정량적 분석이라는 면에서 매우 유용하다. 그러나, 실시간 처리와 같은 시간의 제한적 요소를 가지고 있는 응용분야의 요구를 완전하게 표현할 수 있지는 못하다.

고 신뢰 실시간 응용분야의 요구사항을 다음과 같이 요약할 수 있다[16]. 첫째 요구사항은 처리 시스템의 의존도가 높아야 된다는 점이다. 시스템의 의존도가 낮다는 것은 자주 결함이 발생하여 서비스가 중단되거나 서비스의 절대 제공시간이 적다는 것을 의미한다. 둘째 요구사항은 실시간 반응시간이 매우 짧아야 된다는 것이다. 실시간 반응시간이 짧다는 것은 시스템의 성능이 우수하여 빨리 처리할 수 있는 것을 의미한다. 세째는 입력원이나 시스템의 오동작 시에도 시스템의 시험을 통한 검증이 가능하여야 한다는 점이다. 첫째 조건인 의존도는 결함의 발생빈도수에 대한 제약을 나타내며 둘째 조건인 반응시간은 결함이 발생하였을 경우에 시스템이 한정된 시간내에 재구성과 복구를 통하여 정상적으로 동작할 가능성에 대한 제약을 나타내며 세째 조건은 외적, 또는 내적 결함에 대한 빠른 검출을 위한 제약조건이다. 이들중 의존도만을 시스템의 요구사항으로 파악하여 시스템을 설계, 또는 분석하는 것은 응용분야의 전체 목적에 맞지 않는 시스템을 설계 또는 분석하는 것과 마찬가지다.

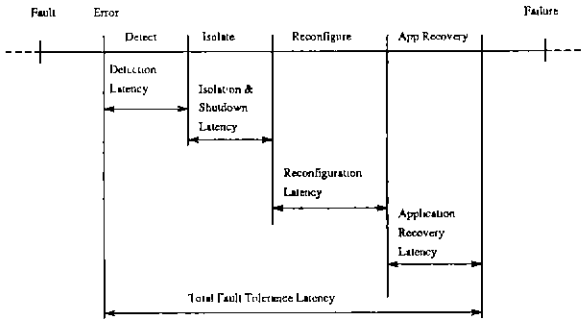
III. 결함포용 지연시간

3.1 개요

결함포용 시스템의 설계에 있어서 중요 관심사중의 하나는 결함포용 지연시간이 응용분야의 요구에 부합하는가에 대한 것이다. 아래 (그림 1)에서 보여준것과 같이 결함포용 지연시간은 오류의 발생으로부터 오류의 복구에 이르기까지 시간 모두를 지칭하는 것이다. 이 지연시간은 시스템이 다음과 같은 행동을 취하는 시간으로 구성되어 있다.

- 오류를 검출 후 위치파악.
- 결함을 가진 module의 고립.
- 재구성, 그리고
- 응용 program을 포함하는 복구 처리시간

본 고에서 언급하는 결함포용 병렬처리 시스템은 복구를 응용 program의 checkpointing과 rollback에 의해 수행한다고 가정한다. 즉, 응용분야 단위의 hot backup이 사용되지 않는다고



(그림 1) 결함포용 지연시간의 정의

가정하는 것을 의미한다. 이는 시스템의 결함포용 지연시간을 비관적인 경우로 분석하고자 하기 때문이다.

3.2 지연시간의 추정

(그림 2)는 지연시간에 관한 문제점을 보여주는 그림이다. 수평축은 결함 또는 오류의 발생으로부터 복구에 이르기까지 응용분야가 참아낼 수 있는 시간을 나타낸다. (그림 2)는 두개의 빗살무늬의 사각형을 가지고 있다. 각각은 결함포용 시스템이 사용되었을 경우 예상 결함포용 지연시간을 나타낸다. 큰 빗금친 사각형은 현재 시스템에서 예견되는 결함포용 지연시간을 나타내며 조그만 빗금친 사각형은 앞으로 5년내지 10년 후의 예상된 결함포용 지연시간을 나타낸다. 그림상의 두 빗금친 사각형은 예견된 성능을 말하는 것으로 미래의 시스템이 가져야 될 성능에 대한 요구사항을 말하는 것은 아니다.

큰 사각형은 결함검출과 고립시간이 약 0~10 milliseconds, 재구성 지연시간이 약 40~90 milliseconds, 그리고 복구지연시간이 약 0~10 milliseconds라고 가정 후 얻어졌다. 이러한 가정에 대한 근거는 다음과 같다.

- 검출과 고립에 대한 상한값은 10 milliseconds로써 약 5 milliseconds 간격으로 각 module간에 자기 module이 결함없이 작동한다는 정보를 교환하는 fail-fast 방법을 사용한다는 가정하에서 얻어졌다. 이 방식외에도 또한 내장형 검사기(Builtin-Test)에 의한 계속 검사, 주기를 가지고 수행하는 주기적 검사(periodic test), 그리고 응용 program에 의한 검출/고립기술도 사용된다고 가정하였다.
- 재구성 지연시간은 다음과 같은 가정하에서 얻어졌다. 복구가 요구될 시, 하나 내지 둘 정도의 시스템 program module을 주 기억장치에 읽어 들여야 한다고 가정하자. 각 module의 크기는 약 16 K에서 32 K byte라고 가정하자. 이러한 module을 읽어 들이는데 약 40~90 milliseconds가 필요하다. 이 수치는 현재 기술로 16에서 32 K 정도의 자료를 읽어 들이는데 필요한 통신속도라고 가정하면 된다.
- 복구 지연시간은 기껏 10 millisecond라고 추정하였다. 이유는 많은 실시간 응용 program의 수행시간이 10 millisecond 이내이기 때문이다.

작은 사각형은 앞으로 5년내지 10년 후의 결함포용 지연시간을 예측하여 나타낸 것이다. 그 추측은 다음과 같은 가정하에서 만들어 졌다.

- Module의 처리속도는 기술의 발전에 따라 더욱 빨라질 것이다. 이전의 발전속도를 보아 5년 내지 10년 후의 처리속도는 현재 처리속도의 50 내지 100배의 속도증가가 있을 것으로 예상된다. 병렬처리의 process간의 동기화 문제를 고려하여 그 속도증가가 반정도로 감소하리라 예상하여도 약 25 내지 50배의 속도증가가 예상된다.
- 통신 protocol을 예상하지 않더라도 module간의 통신속도는 5년 내지 10년후에 약 10~30배 정도 빨라질 것이다. 이러한 빨라짐이 정보의 전달속도를 빠르게 할 것이다.

위에서 하나 주목하여야 할 것은 각 module에서 현재 수행되는 software가 5년 내지 10년 후에 계속하여 동일한 software가 수행된다는 가정하에서 위의 분석이 이루어졌다. 추정하건데, 처리속도와 기억장치 능력의 증가가 좀 더 많은 software를 하나의 module안에서 수행이 가능하도록 할 것이다. 이것의 영향은 조그만 사각형의 오른쪽편이 좀 더 오른쪽으로 이동하는 것

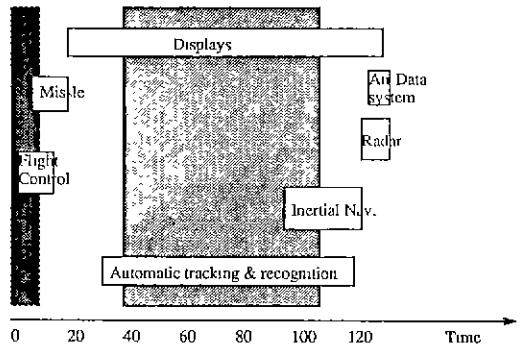
을 의미하는 것이다.

3.3 지연시간의 연계

(그림 2)는 몇몇 종류의 응용분야에 대하여 참아낼 수 있는 지연시간을 나타내는 사각형들이 포함되어 있다. 이들은 주로 실시간 처리 응용분야에서 볼 수 있는 task들로써 sensor에 의해 읽혀진 자료를 처리한다. 이 사각형들은 응용분야에서 받아들일 수 있는 결함포용 지연시간에 대한 요구를 나타낸 것이다. 예를 들어 항공기는 매 40~100 milliseconds마다 새로 측정된 위치, 고도 입력을 받아 공간상의 위치를 수정한다. 그러나 결함에 의한 지연시간이 100 milliseconds 이상이 될 경우 입력 자료를 손실하여 현재 위치에 대한 부정확한 정보를 보유할 수가 있다. 즉, 이러한 응용분야의 결함포용 지연시간에 대한 요구는 40 milliseconds 이하이다.

(그림 2)에서 우리는 추정된 현재의 결함포용 지연시간이 몇몇 응용분야의 요구사항을 만족시키지 못하고 있는 것을 알 수 있다. 즉, checkpointing/rollback을 사용하여 시스템을 운용할 경우 응용분야에서 요구하는 결함포용 지연시간을 만족시킬 수 없다는 것을 의미한다. 이는 NMR이나 hot backup과 같은 기술을 이용하여 결함포용 지연시간이 응용분야의 요구에 맞도록 변경될 것을 요구한다.

같은 시스템내에서 수행되는 task간에 연계성이 존재할 때 요구되는 결함포용 지연시간은 틀려지며 좀 더 작아진다. 다시 말하면, 하나의 task의 결과가 다른 task의 입력으로 사용되는 경우 요구되는 결함포용 지연시간은 단순하지가 않다. 예를 들어, A라는 task는 B와 C라는 task로부터 입력을 받고 B와 C task들은 500 milliseconds의 결함포용 지연시간을 요구하며 A는 350 milliseconds의 결함포용 지연시간을 요구한다고 하자. 만약 B와 C task들이 매 250 milliseconds마다 결과를 만들어 낸다고 하면 B와 C는 A에 자료를 제공함으로, 결함포용 지연시간이 A의 결함포용 지연시간안에 끝나야 한다. A가 50 milliseconds 안에 복구할 수 있다고 하고 또 그것의 출력은 다른것에 의해 사용되지 않는다고



(그림 2) 추정된 결함포용 지연시간

가정하면 A는 350−50=300 milliseconds의 결함포용 지연시간을 가지게 된다. 따라서 B와 C는 300 milliseconds의 결함포용 지연시간 요구를 가져야만 한다. 만약, B와 C에 대한 checkpointing이 되어 있지 않을 경우 250 milliseconds의 복구시간을 B와 C가 요구한다고 하면 오직 50 milliseconds만이 검색, 고립, 재구성하는데 가능한 시간이 된다.

결론적으로 결함포용 지연시간은 응용분야가 사용하고자 하는 시스템에 대하여 적합한지를 판단하는 데 사용되며, 또한 결함포용 방법을 응용분야의 요구에 맞게 선택하는 데에도 사용된다.

IV. 지연시간 분석모델

본 장에서는 지연시간 분석에 관련된 모델을 소개하고자 한다. 결함포용 지연시간의 분석은 결함의 검출로부터 응용 program의 복구에 이르기까지의 과정을 하나의 모델로 표현 분석하는 접근방식과 결함포용의 각 단계별 과정을 모델화하여 나중에 전체모델을 구성하는 접근방식이 있다. 앞에 언급된 방식은 검출로부터 복구에 이르기까지의 과정이 NMR 방식처럼 단순한 경우에는 가능하나 결함검출, 고립, 재구성, 복구의 일련의 과정을 걸쳐 결함포용을 하는 경우에는 각 단계가 각각의 분석모델을 사용하여야 분석이 가능하다. 본 고에서는 각 단계별 분석모델과 그 연구 현황을 살펴보고자 한다.

4.1 Fault set 정의

Fault set의 정의는 fault의 검출방법과 그에 따른 지연시간과 직접 연관되므로 매우 중요하다. 현실에 있어서 모든 결함을 포용할 수 있는 시스템을 설계한다는 것은 거의 불가능한 일이다. 따라서 시스템에서 포용할 수 있는 결함의 종류를 정의하고 그에 대한 지연시간을 구하는 것이 합리적인 접근 방식이다.

병렬처리 시스템에서의 결함에 대한 연구는 매우 드물다. 하나의 방식은 각 node의 기능을 computation과 communication의 두가지로 구분한 후 이들에게서 발생할 수 있는 결함을 세가지로 구분하는 방식이다[17]. 첫째 유형의 결함은 computation기능이 손실되는 결함이다. 이러한 결함의 발생시 communication 능력은 계속 살아 있음으로 시스템은 통신 structure에 대한 재구성은 필요로 하지 않을 수도 있다. 즉 결함이 발생한 node에서 수행중이던 process들만을 다른 node로 옮김으로써 복구를 할 수 있다. 둘째 communication 기능이 손실되는 결함이다. 이러한 경우에는 communication structure의 재구성을 필요로 한다. 세번째는 computation과 communication 기능을 동시에 손실하는 결함이다. 이러한 경우에는 communication 기능이 손실되었을 경우와 마찬가지로 취급한다.

4.2 검출 지연시간 분석

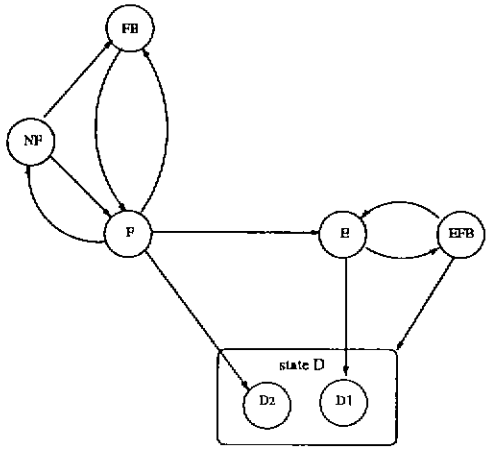
이 절에서는 검출 지연시간과 그의 분석모델에 대하여 알아보기로 하겠다. (그림 1)에서 보여준 것과 같이 검출 지연시간이라 함은 결함의 검출 지연시간을 지칭하는 것이 아니라 오류(error)의 검출 지연시간을 뜻한다[18, 19]. 검출지연시간은 피해 산정, 복구, 그리고 결과에 대한 확신을 가지기 위해서 필요하다. Courtois[18]는 M6800 microprocessor의 on-line test에 의한 결과를 발표하였는데 이중 일련의 검출방법에 대한 검출시간의 분포를 포함하였다. Shedlestsky[19]는 결함 집합과 입력신호의 확률분포에 기초로 한 검출 지연시간을 평가할 수 있는 방법을 제안하였다.

검출 지연시간이 클 때 시스템이 틀린 계산결과를 출력할 가능성이 있다. 이는 출력 당시에 검출되지 않은 오류가 존재할 수도 있고, 출력 이전에 모든 오류가 검출되었다 할지라도 오류의 잠복기간동안 얻어진 결과가 이미 오염되었을 수도 있기 때문이다. 현실적으로 검출 지연시간이 0이 될 수는 없다. 또한 오류가 발생한 경우에는 검출 지연시간 동안 번진 오염을 제거하기 위하여 좀더 복잡한 복구를 하느라 전체 시스템의 수행이 지연될 수도 있다.

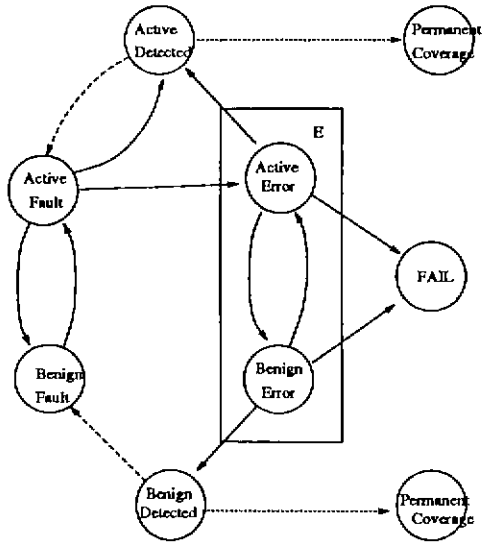
현재 알려진 지연시간 분석모델에 관한 연구는 오류 검출과정 분석모델[5]과 결함상태 전이모델[6]이 있다. 오류 검출과정 분석모델[5]에 따르면 오류는 3가지 방법에 의하여 검출이 가능하다. 이들은 신호 level 검출(Chip-level Built-in Test), 기능 level 검출(System-level Built-in Test), 그리고 주기적 진단(Software Test) 등이다. 신호 level 검출이라 함은 chip의 설계 제작시 fault로 발생된 오류를 검출할 수 있는 self-checking회로를 첨부하는 방식이다. 이 방식의 장점은 fault의 신속한 검출로 오류로 인한 오류발생범위(contamination)가 작다는 것이다. 단점은 부수적 회로(검출을 위한)로 인한 제작경비의 상승과, 모든 결함을 완전하게 검출할 수 없다는데 있다. 기능 level검출이라 함은 신호 level의 윗단계에서, 포용될 수 없는 activity나 information을 검사함으로써 오류를 검출한다. 이 검출방식의 중요한 문제점의 하나는 오류의 고립과, 피해산정에 있다. 세번째로 주기적 진단이라 함은 system을 offline하여 진단 program을 수행하여 결함부위를 검출하는 방식이다.

[5]에서 제안된 분석모델은 결함과 오류를 여러단계로 분류하고 단계간의 전이를 Markov chain으로 나타내었다. (그림 3)은 Markov chain으로 나타낸 오류 검출과정 모델이다[5]. 단계간의 전이율은 위에서 언급된 오류 검출방법을 사용하여 구하도록 하였다. 그림 상에서 각 state가 나타내는 의미는 다음과 같다.

1. NF(Non Faulty) 결함이 존재하지 않은 상태를 말함
2. F(Faulty) 결함이 존재하나 아직 오류를 야



(그림 3) 오류검출과정의 모델



(그림 4) CARE III의 FEHM

기시키지 않은 상태.

3. FB(Fault-Benigh) 반복 결함(Intermittent fault)이 현재 결함을 보이지 않고 있는 상태

4. E(Error) 적어도 하나의 검출되지 않은 오류가 존재하고 결함은 아직도 존재하는 상태.

5. EFB(Error-Fault-Benigh) 오류가 현재 존재하고 있으나 오류를 발생시킨 원인인 일시결함이 사라졌거나 또는 반복결함을 보이지 않고 있는 상태.

6. D(Detection) 오류가 검출방법에 의해 감지된 상태. 감염 영역에 따라 상태를 둘로 세분한다. 결함발생후 바로 검출되었을 경우 D2 상태로, 오류에 의한 감염영역이 있을 경우 D1 상태로 표시한다.

그림에서 state F에서 state D2로의 전이는 주기적 진단 program에 의한 검출을 뜻함으로 전이율은 진단 prorgam의 수행주기에 의존된다. 시스템이 일단 state E로 전이하게 되면 잘못된 오류정보는 기능검출 방식이 받아들일 수 없는 결과를 감지할때까지 계속 번져간다. State E에서 D1으로의 전이는 결함이 계속해서 존재하는 경우임으로 신호 level 검출이 가능하다. State 간의 모든 전이율을 구하고 난 후에 그림의 Markov chain의 해를 구함으로써 검출 지연시간의 분포를 구할 수 있다.

또 다른 모델인 결함상태 전이모델[6]은 신뢰도 분석을 위해 만들어진, 수학 Package CARE III에서 지원되는 분석중 일부분인 coverage를 구하기 위하여 제안된 모델이다. CARE III의 사용자는 모델하고자 하는 시스템을 행태(behavior)에 따라 구분된 두 종류의 입력을 하여야 하는데 이들은 FORM(Fault Occurence and Repair Model)과 FEHM(Fault and Error Handing Medel)이라고 불리운다. 이중 FEHM model은 coverage을 구하기 위하여 사용된 모델로써 (그림 4)은 FEHM model의 state간 상태 전이도를 보여준다. 각 state간의 상태전이는 확률로 표시하게 되어 있다. State간의 전이확률을 서로 다르게 표시함으로써 일시, 반복, 영구결함 등을 표현할 수 있다. 검출지연시간은 결함발생으로 인한 상태전이가 발생할 때의 검출확률을 구함으로써 검출 지연시간을 추정할 수 있다.

CARE III에서 사용된 상태전이 모델[6]은 오류 검출전이모델[5]과 상당히 유사한 점이 있다. 그러나 오류검출 전이모델에서는 상태간 전이율을 결함의 발생율과 검출방법을 모델화하여 전이율을 연속시간 한정상태(continuous-time finite state)의 정지 프로세스(stationary process)로 분석한 반면 상태전이 모델은 discrete-time 한정상태(finite-state)의 정지 프로세스로 분석하였다는 차이점이 있다.

4.3 고립 지연시간 분석

고립은 faulty module의 위치파악과 분리를 말한다. 결함부위의 위치파악은 일반적으로 결함검출 프로세스에 의해 이루어진다. 그러나 오류의 검출이 발생으로부터 많은 지연시간을 갖고난 후에 검출이 되었다고 하면 오류로부터 위치의 파악은 대단히 힘들다. 진단 program의 수행으로 위치파악을 하고자 할 경우에는 진단 program의 수행시간이 곧 위치파악을 위한 지연시간이 될 것이다.

위치파악에 의한 지연시간을 줄이고자 중복되는 부분들을 많이 보유하고 있는 시스템에서는 기능별 위치파악과 고립을 수행한다. 기능별 위치파악이라 함은 어느 하나의 기능을 수행하는 module이 그 기능을 정상적으로 보여주는가를 판별하고 고립을 추진하는 방법이다. 즉 processor board에서 검출되었을 경우에는 board내의 결함부품의 위치를 파악하는 대신에 processor board 하나를 전체 시스템에서 고립하여야 될 부위라고 결정한다.

결함으로 인한 영향이 계속 번져나간 다른 정상적인 시스템에 영향을 주는 것을 막기 위해 결함의 영향을 한 곳에 국한(confinement)하는 것이 필요하다. 결함이 계속 영향을 주는 것을 막기 위한 하나의 방법으로 결함이 있는 module이 다른 건강한 module을 access하는 것을 막아야 한다. 이러한 역할은 운영체제에서 지원하여야 될 사항이다.

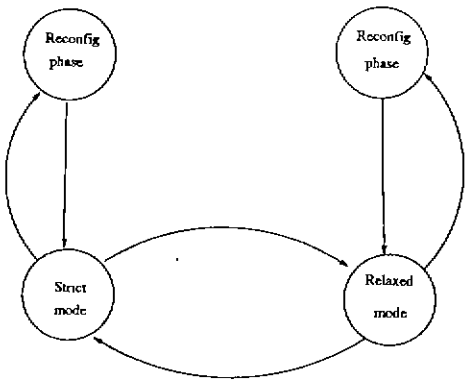
결함의 위치파악, 고립, 영향국한에 의한 지연시간에 대한 연구는 따로 보고된 적은 없다. 이는 위치파악은 일반적으로 검출의 일부분으로 파악하고 있기 때문이다. 또한 고립과 영향의 국한과 같은 목적은 기능 level의 결함검출과 고립으로써 만족시킬 수 있고 이에 소요되는 지연시간은 매우 미미하다고 여겨지기 때문이다.

4.4 재구성 지연시간 분석

결함의 검출과 위치파악, 그리고 고립후의 단계는 시스템의 재구성이다. 재구성이라함은 결함없는 부분들을 이용하여 시스템이 최적의 성능을 낼 수 있는 구조로 변환하는 것이다. 여기서 구조변환이란 물리적인 구조변화를 뜻하는 것이 아니라 task들을 배치함에 있어서 성능이나 신뢰도를 최적으로 가질 수 있는 software와 hardware의 통합구조의 변환을 말한다.

재구성 지연시간에 관련되어 세가지 방향의 연구가 진행되고 있다[7-9]. 첫번째 연구방향은 재구성될 구조의 선택방법에 관한 연구이다[7]. 이 연구에서 시스템의 재구성방식을 동적(dynamic) 재구성방식과 정적(static) 재구성방식으로 구분하였다. 정적 재구성방식은 시스템의 결함발생전 가능한 재구성 구조를 판단 후 Table을 구성하여 놓은 후 결함발생시 table에서 가장 적합한 구조를 선택하는 방식이다. 동적 재구성방식은 결함발생시 활용할 수 있는 자원(resources)과 부하량(workload)을 분석하여 최적의 구조를 선택하는 방법이다. 동적 재구성 방식은 결함발생시 가장 적합한 재구성 구조를 선택할 수는 있으나 재구성 지연시간이 정적 재구성 방식보다 많이 요구된다는 결점이 있다. 정적 재구성방식은 재구성 지연시간을 단축시킬 수 있으나 결함발생시 최적의 구조가 table에 포함되어 있지 않은 경우 재구성 구조가 최적성을 같지 못한다는 결점이 있다.

두번째 연구방향은 재구성 구조선택시 환경에 의한 재구성 구조 선택방법에 관한 연구이다[8]. 이 연구에서 시스템에 결함이 발생하였을 때 시스템의 상태를 두가지로 구분하였다. 그들은 "strict mode"와 "relaxed mode"이다. Strict mode는 많은 계산부하와 한정시간내에 반응이 있어야 된다는 요구 등으로 특징지울 수 있으며 relaxed mode는 적은 계산부하와 완화된 반응시간 제약 등으로 특징지울 수 있다. 시스템은 두가지 상태 사이를 순차적으로 전이되나, 그 전이주기는 일정한 것으로 정해졌거나 예측할 수 있는 것은 아니다 각 상태에서 결함이 발생하였을 시 그들의 재구성 구조 선택방법에는 차이가 있을 수 있다. Strict mode에서는 반응시간에 대한 시간적 제약이 있음으로 재구성구조를 선택할 시 국지적 정보를 이용 빠른 시간내에 재구성을 완료하여야 한다. 물론, 재구성된 구조는 최적의 구조가 아닐 수도 있다. 반면, relaxed



(그림 5) 이중상태에서의 재구성 전이모델

mode에서는 시간에 대한 제약이 없으므로 많은 시간이 걸릴지라도 최적의 구조를 선택한다. (그림 5)는 두가지 상태에서의 재구성에 관한 전이모델을 보여준다[8]. 이러한 방식의 재구성 구조 선택은 시스템의 반응시간 요구사항에 부합하여 작동할 뿐만 아니라 시스템의 작동시간을 연장시켜주는 장점도 있다.

세번째 연구방향은 재구성의 수행에 관한 연구이다[9]. 이 연구에서는 시스템의 재구성을 수동적 재구성과 능동적 재구성으로 구분하였다. 수동적 재구성이라 함은 결함의 발생 등과 같이 내부구성을 바꾸어야 하는 사건이 발생하였을 시에 재구성을 수행하는 것을 말한다. 이에 반하여 능동적 재구성이라 함은 결함의 발생과 같은 외적 조건의 발생이전에 시스템의 상태를 평가하여 최적의 구조로 재구성을 수행하는 것이다. 시스템의 조건 평가는 성능과 신뢰도를 합한 성능신뢰도(performability)를 사용하였다.

시스템 재구성과 관련 하나 생각하여야 될 문제는 재구성구조의 선택기준이다. 최적(optimal)의 선택은 선택기준이 무엇이었는가에 따라 달라진다. 재구성구조의 선택기준으로써 세가지를 들 수 있다. 그 첫째가 성능(performance)을 선택기준으로 삼는 것이다. 즉 재구성될 수 있는 구조 중에서 성능이 우수한 구조를 선택하는 것이다. 여기서 성능이라 함은 예를 들어 processing power 또는 잔존 수행시간(remaning mission time) 등을 꼽을 수 있다. 두번째 선택기준으로써 성능신뢰도 (performability)[20-22]을 들 수 있다. 즉 선택기준을 성능 또는 신뢰도 한 측면만을 고려하지 않고 성능과 신뢰도를 복합하여 보는 것이다. 성능신뢰도 방식은 성능과 신뢰도를 동시에 비교, 분석한다는 잇점은 있으나 이의 계산이 상당히 복잡하여 재구성 지연시간이 증가한다는 단점이 있다.

4.5 복구 지연시간 분석

시스템의 재구성 이후 응용 program의 복구가 필요하다. 복구과정은 각각의 module에 process들을 loading하고 backup이나 checkpointing에 의해 저장된 결함시점으로부터 가장 가까운 시점의 정보를 읽어 결함발생전의 환경으로 복구한다. Checkpointing이나 backup을 자주 할 경우에는 checkpointing 시점과 결함발생 시점과의 간격이 작음으로, 따라서 복구 지연시간도 작다. 그러나 자주 checkpointing을 함으로써 전체적으로 process의 수행에 지연이 발생한다. 따라서 적절한 간격을 선택하여야만이 복구시간을 단축시킬 수 있고 정상상태에서도 checkpointing에 의한 부하를 줄일 수 있다. 적절한 checkpointing 간격에 관한 연구는 많은 발표가 있었다 [10-12].

복구 지연시간은 전적으로 checkpoint의 삽입간격에 좌우된다. 그러나 병렬처리 컴퓨터에 있어서 checkpoint와 process간의 통신지점의 잘 조화되지 않은 삽입은 rollback recovery point가 뒤로 후퇴하는 rollback propagation(domino 효과라고 함) 결과를 가져온다. 이는 곧 복구 지연시간이 증가하는 효과를 가져온다. Domino 효과를 방지하기 위하여 두가지 방법을 일반적으로 사용한다. 첫째는 동기화(synchronous) checkpoint 방식이다[23]. 동기화 방식은 checkpoint의 삽입지점과 process간의 통신지점을 잘 조화시켜 복구지점이 뒤로 후퇴하는 domino 효과를 막는 방식이다. 동기화 방식의 실예로는 rollabck propagation detection방식[24], 중복된 recovery point 삽입[25], two-phase commit protocol[23], 그리고 pseudo recovery point 등이 있다. 이 방식의 단점은 정상동작 중에 조화있는 checkpoint의 삽입지점을 찾기위해 사용되

는 부수적인 부하이다. 두번째 방식은 자료 저장 방식으로 process간의 통신 message을 저장하여 rollback propagation recovery시에 저장된 message를 다시 복원함으로써 복구지점이 뒤로 후퇴함에 따른 피해를 최소화하는 방식이다[26].

위에서 언급된 rollback propagation외에 또 다른 하나의 문제가 있다. 이는 error propagation이다[27]. 즉 완전치 못한 오류의 검출 또는 지연된 오류의 검출과 process간의 통신에 의한 결과로 checkpoint된 정보를 신뢰할 수 없어 이전의 checkpoint된 상태까지 rollback하여야 된다. 이러한 경우에 있어서 가장 중요한 문제는 오류에 의한 피해 산정(damage assessment)에 있다. 정확한 피해 산정만이 올바른 정보만을 가진 checkpoint까지 후퇴가 가능할 것이다.

V. 결 언

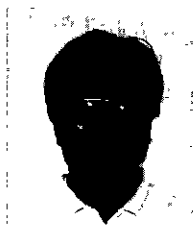
본 고에서는 결함에 의한 오류의 발생으로부터 복구가 이르기까지 소용되는 결함포용 지연시간 분석모델에 대하여 알아 보았다. 결함포용 지연시간 분석은 발생되는 fault set의 정의에서부터 시작한다. 그 다음으로 결함포용의 각 단계에서의 지연시간분석과 그들을 다 모은 시스템 수준에서의 지연시간 분석을 요구한다. Fault set에 대한 정의는 검출지연 시간이 전적으로 정의된 Fault set의 발생과 검출에 의존한다는 면에서 매우 중요하다. 검출과 고립지연 시간 분석모델은 정의된 fault의 상태와 상태별 전이율, 그리고 검출방식에 따른 지연시간 분석이 요구된다. 재구성 지연시간 분석을 위하여 여러가지 재구성 방식에 대해서 알아 보았다. 그들은 최종적으로 재구성구조의 최적성(optimality)에 대한 연구를 요구한다. 재구성구조의 최적성에 대한 연구는 현재 많은 병렬 또는 다중처리기가 재구성을 포용하는 쪽으로 설계되고 있는 추세이므로 매우 중요하다. 복구 지연시간은 checkpointing interval에 전적으로 의존되며 이의 최적화가 필요하다. 복구지연시간에 영향을 미치는 요소로서 recovery propagation과 error propagation에 대하여 알아보았다.

결함포용 지연시간 분석은 앞으로 설계되는 시스템의 응용분야를 쉽게 결정하여 주는 잇점이 있다. 만약에 다중 또는 병렬처리기가 어떠한 응용분야를 염두에 두고 설계 제작되었을 때 시스템의 완성이후 그 구조가 응용 분야의 결함포용 요구도를 만족시킬 수 없다고 한다면 이는 인력과 자원의 낭비와 위험한 결과를 가져올 수 있다. 따라서 제작이전에 이러한 분석이 가능하다면 설계시 이를 반영함으로써 경비를 절감할 수 있다.

참 고 문 헌

1. P. Nelson and B. D. Carroll, *Tutorial: Fault-Tolerant Computing,* IEEE Computer Society Press, 1987.
2. B. W. Johnson, *Design and analysis of fault-tolerant digital systems,* Addison-Wesley Pub. Co., 1989.
3. A. L. Hopkins, T. B. Smith, and J. H. Lala, "FTMP-A highly reliable fault-tolerant multiprocessor for aircraft," *IEEE Proceedings,* vol. 66, pp. 1221~1239, October 1978.
4. D. P. Siewiorek, "Fault tolerance in commercial computers," *IEEE Computer,* pp. 26~37, July 1990.
5. K. G. Shin and Y. H. Lee, "Error detection process-model, design, and its impact on computer performance," *IEEE Trans. on Computers,* vol. C-33, pp. 529~540, June 1984.
6. J. J. Stiffler and L. A. Bryant, "CARE III Phase III report-Mathematical Description," Contract Report 3566, NASA, November 1982.
7. D. Paul, C. Roark, and D. Struble, "Technical rport on phase one of the dynamic reconfiguration demonstration system program," Technical Report NAWC-DRDS-P1-TR-0003, Texas Instrument, April 1992.
8. R. G. Melhem, "Bi-level reconfigurations of fault tolerant arrays in bi-modal computational environments," in *Proc. of 19th International Symposium on Fault-Tolerant Computing,* pp. 488~495, 1989.
9. Y.-H. Lee and K. G. Shin, "Optimal reconfi-

guration strategy for a degradable multimodule computing system," *J. of ACM,* vol. 34, no. 2, pp. 326~348, April 1987.
10. G. Copeland and T. Keller, "A comparison of high availability media recovery techniques," in *2989 SIGMOD Proceedings,* pp. 98~109, 1989.
11. K. M. Chandy, J. C. Browne, *et al.,* "Analytic models for rollback and recovery strategies in database systems," *IEEE Trans. on Software Engineering,* vol. SE-1, no. 3, pp. 100~110, April 1977.
12. E. Gelenbe, "On the optimum checkpointing interval," *Journal of ACM,* vol. 26, pp. 259~270, April 1979.
13. J. C. Laprie and A. Coste, "Dependability: A unifying concept for reliable computing," on *Proc. of FTCS-12,* pp. 18~21, June 1982.
14. D. P. Siewiorek and R. S. Swartz, *Reliable System Design: The theory and practice,* Digital Press, New York, 1992.
15. C. R. Das, J. T. Kreulen, *et al.,* "Dependability modeling for multiprocessors," *IEEE Computer,* pp. 7~19, October 1990.
16. J. H. Lala *et al.,* "A design approach for ultra-reliable realtime systems," *IEEE Computer,* pp. 12~28, May 1991.
17. J. Hastad, T. Leighton, and M. Newman, "Reconfiguring a hypercube in the presence of faults," in *Proc. 28th Annual Symp. on Foundations Computer Sci.,* pp. 274~284, October 1987.
18. B. Courtois, "Some results about the efficiency of simple mechanisms for the detection of microcomputer malfunction," in *Proc. 9th Annu. Int. Symp. on Fault-Tolerant Computing,* pp. 71~74, 1979.
19. J. Shedletsky, "A rollback interval for networks with an imperfect self-checking property," *IEEE Trans. Comput.,* vol. C-27, pp. 500~508, June 1978.
20. M. D. Beaudry, "Perfomance related reliability measures for computing systems," *IEEE Trans. on Computers,* vol. C-27, pp. 540~547. January 1978.
21. C. R. Das and L. N. Bhuyan, "Bandwidth availability of multiple-bus multiprocessors," *IEEE Trans. on Computers,* pp. 918~926, October 1985. Special Issue on Parallel Processing.
22. J. Kim, K. G. Shin, and C. R. Das, "Performability evaluation of gracefully degradable hypercube multicomputers," in *1992 IEEE Workshop on Fault-Tolerant Parallel and Distributed Systems,* pp. 140~147, July 19 92.
23. W. H. Kohler, "A survey of techniques for synchronization and recovery in decentralized computer systems," *Computing Surveys,* vol. 13, pp. 149~183, Junme 1981.
24. K. Tsuruoka, A. Kaneko, and Y. Nishihara, "Dynamic recovery schemes for distributed processes," in *Proc. IEEE Reliability in Distributed Software and Database Systems,* pp. 124~130, July 1981.
25. K. Kant and A. Silberschatz, "Error recovery in concurrent processes," in *Proc. COMPSAC,* pp. 608~614. 1980.
26. R. E. Strom and S. Yemini, "Optimistic recovery in distributed systems," *ACM Trans. Computer Systems,* vol. 3, pp. 204~226, August 1985.
27. K. G. Shin and T.-H. Lin, "Modeling and measurement of error propagation in a multimodule computing system," *IEEE Trans. on Computers,* vol. 37, no. 9, pp. 1053~1066, September 1988.



김 종

1981 한양대학교 전자공학과(학사)
1983 한국과학기술원 전산학과(석사)
1991 Pennsylvania State, University 전산기 공학(박사)
1983 ~1986 한국중공진산 System programmer
1991 ~1992 University of Michigan 연구원
1992 ~현재 포항공과대학 조교수
관심 분야 : 결함포용, 병렬처리, 성능분석