論文94-31B-8-1

缺陷許容 분산시스템의 재분배 알고리즘의 시뮬레이션과 평가

(Simulation and Evaluation of Redistribution Algorithms In Fault-Tolerant Distributed System)

崔 柄 甲*, 李 天 熙**

(Byung Kab Choi and Cheon Hee Yi)

要 約

본 논문에서는 분산 시스템에서 n개의 노드에 결함이 발생했을때 결함 노드 의 부하 재분배를 통하여 결함을 허용하는 부하재분배 알고리즘들을 설계하였다. 각 알고리즘들에 대한 효율성 평가를 위하여 시뮬레이션 모델을 구축하고 작업 도착율,서비스율,노드의 고장율과 수리율,부하 이주에 따른 통신지연을 매개변수로 하여 시뮬레이션 한 결과 작업 도착율,서비스율,수리율은 알고리즘들간의 상대적 효율성에 큰 영향을 주지 않았으나 통신지연시간의 크기는 커다란 영향을 미침을 알았다. 통신지연시간이 평균 작업 처리시간보다 클때는 알고리즘 B가 우수했고 비슷하거나 작을 경우는 알고리즘 C가 현격히 우수함을 보였다.

Abstract

In this paper,load redistribution algorithm to allow fault-tolerance by redistributing the workload of n failure nodes to the remaining good nodes in distributed systems are investigated. To evaluate the efficiency of the algorithms, a simulation model of algorithms is developed using SLAM II simulation language. The job arrival rate,service rate,failure and repair rate of nodes,and communication delay time due to load migration are used as parameters. The result of the simulation shows that the job arrival rate, failure and repair rate of nodes do not affected on the relative efficiency of algorithms.If the communication delay time is greater than average job processing time,algorithm B is better. Otherwise,algorithm C is superior to the others.

*正會員, 牧園大學校 理工大學 컴퓨터工學科
(Dept. of Computer Eng, MOKWON Univ.)
**正會員, 淸州大學校 電子工學科
(Dept. of Elec. Eng., Cheongju Univ.)
接受日字 : 1993年 10月 21日

I. 서 론

최근 C & C의 발전으로 LAN으로 구성되는 다운사이징의 분산 시스템 구축이 확산됨에 따라 자원의 공유와 병행처리를 제공하는 분산 처리 시스템의 구축이 일반화되고 있어 부하 재분배를 통한 결함허용

기법이 점차 중요시 되고 있다. 그러나 기존의 결함 허용을 위한 재분배 알고리즘의 연구는 해석적 방법으로 접근하였기 때문에 분석에 많은 제약이 따랐다. 단순화를 통한 기존의 해석적 연구방법은 분산된 노드들간 통신비용을 고려하지 못했을 뿐만 아니라 재분배 알고리즘들에 대한 효율성 평가보다는 안정성이라는 간접적 평가뿐이었다.[1,2]

본 논문에서는 분산 시스템의 노드 결함시 결함허용을 위한 부하 재분배 알고리즘의 시뮬레이션을 통하여 보다 실제적인 분석을 하고자 한다.[3] 분산 시스템에서 n개의 노드에 결함이 발생했을때 결함 노드 부하의 재분배 여부로 결함을 허용하는 부하재분배 알고리즘을 모델링하였다. 이때 부하재분배는 모든 노드들의 부하가 균형을 이루도록 정상 노드들의 부하상태를 고려하였고 각 알고리즘들에 대한 평가를 위하여 시스템성능의 평가지표를 산출할 수 있는 작업 도착율, 서비스율, 노드의 고장율과 수리율,부하 이주에 따른 통신지연등을 매개변수로하여 시뮬레이션 하였다. 2장에서는 결함허용기법들을 개관하고 결함허용을 위한 부하 재분배 알고리즘모델을 설계한다. 3장에서는 각 알고리즘에 대한 시뮬레이션 프로그램을 작성하고 4장에서는 시뮬레이션의 결과를 분석하여 재분배 알고리즘의 효율성을 평가하고 끝으로 5장에서는 결과의 시사점을 논의하였다.

Ⅱ. 결함허용방식과 결함노드의 부하재분배

분산 시스템은 결함 노드의 작업부하를 정상노드나 예비노드로 전송하여 실행을 계속 할 수 있으므로 단일 프로세서 시스템에 비하여 본질적으로 보다 높은 가용성과 신뢰성을 제공한다. 따라서 fail-soft 능력을 갖는 분산 시스템을 구현하기 위하여는 결함 노드의 작업 부하를 재분배 하는 알고리즘이 필요하다. 결함 노드의 작업 부하를 재분배함으로써 시스템의 하나 또는 그 이상의 정상 노드들이 과부하 상태가 되어 심각한 시스템 성능의 저하를 초래 하는 불안정 상태를 유발할 수도 있으므로 재분배 알고리즘은 신중하게 설계,사용되어야 한다. 이 때문에 결함하에서 분산 시스템의 동작과 성능에 대한 연구의 관심이 증대되고 있다.[4,5]

초기의 결함허용기법의 주종은 중복방법이었기 때문에 지나친 비용의 부담으로 전자 교환기, 핵 발전 제어, 우주 산업,항공 산업 등 신뢰도가 절대로 요구되는 특수한 영역에서만 적용하였다.[6,7,8] 그러나 컴퓨터 기술의 발전과 응용 분야의 확대로 공장,병원, 은행,교통 제어,사무 분야 등에서도 신뢰도가 높은 컴퓨팅 시스템이 절실하게 요구되어 결함을 허용하는 컴퓨터까지 등장하였다.[8,9,10] 기존의 결함허용기법으로는 하드웨어적으로 NMR(N-modular Redundancy),Self-Purging등의 정적 중복방법과 대기중이던 예비모듈로 대체시키는 동적인 중복방법이 주종이었으나 전술한바와 같이 시스템의 비용이 과다하였다. 소프트웨어적으로는 NMR과 유사한 N-Version 프로그래밍기법과 동적 중복방법인 N-Self Checking 프로그래밍기법, 검사점 이용기법,Recovery Block기법등이 있다. 기타 감시 타이머,패리티 비트 검사,Checksum등 여러가지가 있다.[11]

하드웨어 결함시의 공유 메카니즘은 결함이 발생한 노드들을 모두 배제 시키고 정상노드들 만으로 재구성하여 처리한다. 이 방법은 전체 연산능력의 감소를 초래하기 때문에 작업의 도착이 일정하다고 가정하면 평균 응답시간이 늘어지고 대기큐의 길이도 길어지게 되어 시스템의 성능이 심각하게 감소될 수도 있으며 때로는 시스템이 불안정 상태가 될 수도 있다. 결국 결함환경을 고려한 분산 실시간 시스템에서 부하의 재분배는 시스템의 안정성 평가가 반드시 수반 되어야 한다. 본 장에서는 어떤 환경에서 어느 알고리즘들이 사용될 수 있을것인가를 위한 부하 재분배 알고리즘을 설정한다.

1. 재분배 알고리즘 모델을 위한 가정

분산 시스템에 작업들이 도착하면 부하균형 알고리즘은 여러 노드로 균배 하며 결함의 탐지는 별도의 감시 노드가 실행하는것으로 가정한다. 분산 시스템은 고장의 발견과 수리가 용이하고 한 노드(중앙 노드제외)의 결함이 전체 시스템에 영향을 미치지 않으며 확장이 용이한 방사형의 LAN으로 가정하고 재분배 알고리즘을 설계하기 전에 필요한 추가 가정들을 다음과 같이 설정한다.

[가정 1] 시스템 내 각 노드는 새로이 도착하는 작업들을 버퍼링할 수 있는 큐를 갖고 작업의 이주를 위하여 통신 서브 시스템에 라인 큐를 갖는다고 가정한다.

[가정 2] 결함 노드의 갯수는 임의개 발생할 수있다.

[가정 3] 작업의 도착율(λ),결함 발생율,작업 서비스율,고장 수리율은 임의 의 확율분포이며, 각 노드의 서비스율은 동일하다.

[가정 4] 통신 서브시스템의 통신 전용처리기는 각 노드의 대기큐의 상태정 보를 관리한다. 부하 이주시 노드간의 통신 지연시간(CDT:Communication Delay Time)

은 평균 작업 서비스 시간의 배수로 하고 통신결함은 없다.

[가정 5] 시스템의 성능 측정지표는 작업 대기 큐와 라인 큐의 평균 길이로 한다.

2. 부하 재분배 알고리즘

결함노드의 부하 재분배로 인하여 특정 노드가 과부하 되지 않도록 균등화 되어야 한다. 그런데 노드 결함시 대처할 수 있는 방법으로는 노드가 수리,복구 될때까지 작업들을 대기시키는 방법(이는 부하의 재분배를 통한 결함을 허용하지 않는 경우로 볼 수 있다),다른 정상 노드들에게 재분배하는 방법,끝으로 예비 노드로 이주하여 작업을 계속 실행하는 방법이 사용될 수 있다. 그러나 마지막 방법은 많은 비용이 소요되는 종래의 중복기법이므로 본 논문에서는 고려대상에서 제외하고 복구시간이 통신지연시간에 비해 아주 짧거나 다중 처리기로 구성된 분산 시스템의 경우에 적합한 첫번째 방법과 통신기술의 발전으로 통신지연이 작다고 보아 두번째 방법을 연구 대상으로 하였다. 이 두가지 방법이 함께 사용될 수 있는 방법은 아래 도식과 같다.

	대기 큐에있는 작업들	대 기	재 분 배
새로 도착하는 작업	큐에서 대기	알고리즘A	알고리즘B
	재 분 배	알고리즘D	알고리즘C

여기서 알고리즘 D는 결함전에 대기하던 작업들이 새로 도착하는 작업들보다 처리가 늦어지므로 응답시간이 길어지게 되어 실시간 시스템에서는 바람직한 알고리즘이 못되어 연구대상에서 제외하였다. 이는 (1),(2)에서 사용한 알고리즘들과 동일하다.

[알고리즘 A:부하의 재분배를 실시하지 않는다]

결함이 발생한 노드의 대기 큐에 있는 작업은 물론 수리중에 도착하는 모든 작업들도 버퍼링 하였다가 수리가 완료된 후 재시작 한다. 이 알고리즘은 결함허용을 위하여 결함 노드의 부하를 재분배하지않는 것으로서 짧은 시간내에 수리가 가능 하거나 작업의 이동에 통신 지연이 큰 시스템,또는 분산 시스템에서 배치처리할 경우 등에 적용할 수 있을 것이다.

[알고리즘 B:부분적인 재분배를 한다]

결함이 발생한 노드의 대기 큐에 있는 작업들만 정상 노드들에게 균배하고 수리기간에 도착하는 작업들은 버퍼링 하였다가 수리가 완료된 후 재시작 한다. 이 알고리즘은 단시간 내에 복구가 가능 하거나 통신비용이 무시못할 정도인 경우,또는 치명적인 결과를 초래하지 않을 정도의 소프트 실시간 분산 시스템에서는 고려해 볼 수 있는 알고리즘이다.

[알고리즘 C:모든 작업을 재분배 한다]

결함 노드 대기 큐내의 작업은 물론 수리 중에 도착하는 새로은 작업들도 정상 노드로 재분배 한다. 이는 결함의 발생으로 인하여 치명적인 결과를 초래하는 하드 실시간 분산 시스템이거나 복구가 불가능한 결함인 경우 통신 지연의 대소에 관계없이 결함노드에 할당된 모든 부하는 정상 노드들에게 균배 되어야 한다. 또는 작업의 이주에 따른 통신지연이 별 문제가 없을 정도로 아주 작고 모든 부하를 이주 하더라도 시스템의 안정에 영향이 없다고 판단되는 상황에 적용할 수 있을 것이다.

알고리즘 B와 알고리즘 C를 사용할 경우 작업 큐에 대기중이던 작업들의 재분배는 다음과 같이 3가지 유형이 있을 수 있다.

① 결함노드의 부하를 임의적으로 선택된 한 노드로 일괄 이주시킨다.

② 결함노드의 부하를 모든 정상 노드들에게 균배한다.

③ 결함 노드의 부하를 정상 노드들의 현재 부하상태를 고려하여 재분배 후 에도 정상 노드들의 부하가 가능한 균형되도록 재분배한다.

①,②는 경우에 따라서는 노드들이 과부하되어 시스템이 불안정상태가 될 수있어 본 논문에서는 ③의 경우만을 고려한다. 모든 노드의 현재 부하 상태의 정보를 갖고있는 큐 상태 테이블의 큐 길이 정보를 사용하여 큐 길이가 가장 짧은 노드로 한번에 하나의 작업씩 이주하도록 하는 방법을 취하였다. 이 방법은 작업의 재분배에 상당한 시간이 소요될 수 있다.

Ⅲ. 시뮬레이션 모델의 구조 : SLAM 프로그램

1. 재분배 알고리즘의 효율성 평가를 위한 시뮬레이션 모델

Chou등은 분산 시스템에서 하나의 노드가 결함일때 시스템의 동작을 분석하기 위하여 부하 재분배 알고리즘에 대한 시스템의 안정성 평가를 하였다.[1] 연구결과 알고리즘 A는 매우 안정 알고리즘으로 보일 수 있으나 불안정한 시스템으로 유도될 수 있고 알고리즘 B는 대체로 안정적인 반면 알고리즘C는 노드의 이용율에 좌우된다고 하였다. 그러나 이들의 연구결과를 실제 분산 시스템에 적용하기에는 다음과 같은 문제점이 제기 된다.

① 대기행렬의 길이에 관하여 단정적인 예측을 제시하지 못하고 범위만을 제시 했기 때문에 가능한 대기행렬 길이의 범위가 중첩되는 경우 알고리즘 선택

에 어려움이 따른다.

② 재분배 알고리즘들간의 효율성 평가가 아니라 각 알고리즘의 상대적 안정성 평가에만 국한 하였기 때문에 분산 시스템의 특정 환경에 따라 어떤 알고리즘을 선택할 것인가의 명확한 기준을 제시하지 못한다.

③ 분산 시스템에서 단지 하나의 노드만 결함이 발생한다고 가정하였고 통신의 지연은 전혀 고려하지 않았다. 그러나 결함은 임의개 발생할 수 있으며 분산 시스템에서 통신의 지연은 다중 프로세서 시스템과는 달리 무시할 수 없기 때문에 특히 이들은 중요한 한계점이라 할 수 있다.

본 논문에서는 Chou등의 해석적 연구에 따르는 제약들을 극복하기 위하여 시뮬레이션 접근방법을 사용하였다. 해석적 방법은 복잡한 시스템의 대기행렬 길이에 관하여 개략적인 범위만을 확인할 수 밖에 없는데 반하여 시뮬레이션 방법은 비록 확율적이지만 구체적인 대기행렬 길이를 평가할 수 있기 때문이다.[3] 본 논문에서는 시뮬레이션을 위하여 SLAM II언어를 사용한다.[12] SLAM II는 각종 시뮬레이션 연구에 자주 사용되는 비교적 최근에 개발된 전용 시뮬레이션 언어로써 범용 언어에 비해 모델의 수정과 확장을 용이하게 한다. (그림 1)은 알고리즘 C에 대한 SLAM II 네트워크 모델을 도시한 것으로서 모두 10개의 section으로 구성되었다(SLAM II 프로그램은 지면관계상 알고리즘 C에 대해서만 부록에 보였다.)

(그림 1)에서 분산시스템의 각 처리기들은 자원서버(resource)로 표현되어 있으며, 처리기에 발생되는 고장은 자원서버를 선점(preempt)하는 것으로 표현되어 있다. (그림 1)에서 처리기에 고장이 발생되는 경우 처리중이던 작업 및 대기하던 작업들은 자원(처리기)을 잃어버리게 되며, 여타의 정상노드들로 이주하게 된다. 이때 정상노드로의 재분배는 선입선출법에 의거하여 이루어지며 현재 대기행렬에서 기다리는 작업의 갯수가 가장 적은 정상노드로 재분배된다. 부하의 재분배는 두가지 종류의 시간지연을 발생시킨다. 첫째는 부하균형을 위하여 노드를 탐색하는데 소요되는 시간이며, 둘째는 작업의 전송에 소요되는 시간이다. 여기에서는 이 두가지 종류의 시간지연을 통신지연시간으로 치환하여 시뮬레이션하였다.

2. 재분배알고리즘의 효율성에 영향을 미치는 요인들

실제로 결함허용 시스템을 구축하기 위해서는 먼저 세가지 부하 재분배 알고리즘중 어느 알고리즘을 채택할지를 결정해야한다. 그러나 어느 하나의 부하 재분배 알고리즘이 모든 환경에서 가장 효율적이지는 않다. 그러므로 어느 환경에서 어떤 재분배 알고리즘이 효율적인가에 관한 평가 작업이 필수적이다. 부하 재분배 알고리즘의 효율성은 각 노드의 작업큐의 길이로 측정될 수 있다.[2] 특정의 재분배 알고리즘을 채택하였을 때 각 노드에서 대기하는 작업큐의 길이가 커진다면 이는 작업들을 처리하는데 소요되는 시간이 길어진다는 점을 의미하는데 이는 실시간 시스템에서 문제를 야기시킬 수 있다. 큐의 길이에 영향을 주는 환경적 요인들은 크게 작업의 도착율과 작업의 서비스율로 나눌 수 있다. 그런데 작업의 서비스율은 동일하다고 가정하였기 때문에 본 논문에서는 작업의 도착율에 관련된 요인들을 중심으로 분석하고자 한다.

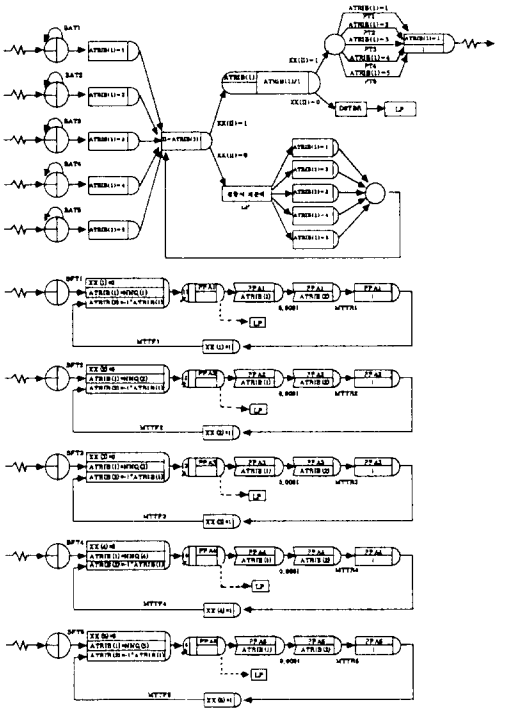

그림 1. 알고리즘 C의 SLAM II 네트워크 모델
Fig. 1. SLAM II Network model of Algorithm C.

결함허용 분산시스템의 경우 각 노드에 도착하는 작업의 유형은 두가지로 구분된다. 첫째는 각 노드의 사용자들로부터 입력되는 내부 작업들이며 둘째는 결함이 발생된 다른 노드로부터 전송된 외부 작업들이다. 결함노드로부터 전송되는 작업의 수에 영향을 미치는 요인으로는 결함발생간격시간, 결함수리시간, 통

신지연시간 등이다. 결함발생간격시간이란 각 노드에 결함이 얼마나 자주 발생되는가를 의미하며 결함수리시간이란 일단 발생된 결함을 수리하는데 얼마나 오랜 시간이 소요되는가를 의미한다. 통신 지연시간이란 결함노드로부터 하나의 작업을 정상 노드로 분배하는데 소요되는 시간을 의미한다.

재분배 알고리즘의 효율성이 큐의 길이에 의해 평가될 수 있기 때문에 큐의 길이에 영향을 주는 요인들은 재분배 알고리즘의 효율성을 결정하는 요인들이라고 할 수 있다. 따라서 재분배알고리즘의 효율성을 결정하는 요인은 결함발생간격시간,결함수리시간,통신지연시간 등의 대소에 의해 구분되어질 수 있다. 결함발생간격시간이 대기행렬의 길이에 미치는 영향은 작업도착율과 유사할 것으로 추정 되므로 본 논문에서는 먼저 결함발생간격시간이 2500인경우 사용자로부터의 작업의 도착율이 재분배 알고리즘의 효율성에 미치는 영향을 검토하고, 결함수리시간 및 통신지연시간이 재분배알고리즘의 효율성에 미치는 영향을 검토한다. 마지막으로 결함발생간격시간을 1500으로 단축시켜 시뮬레이션하여 봄으로써 결함발생간격시간의 영향에 관하여 분석하였는데 이는 CYBER 175와 IBM 360/75 프로세서의 결함율은 $1.4 * 10^{-4}$ faults/min ~ $10 * 10^{-3}$ faults/min 이고 평균 결함율은 $1.4 * 10^{-4}$ faults/min으로 보고된 것을참고 하였다.[1]

3. 시뮬레이션에서 사용되는 파라미터

작업의 처리시간은 모든 처리기에 있어서 동일하게 평균 1의 지수분포를 이룬다고 가정한다. 만약 작업의 도착간격시간이 작업처리시간보다 작아지면 큐길이는 무한히 증가하게 된다. 그러므로 본 논문에서는 작업의 도착율은 작업처리시간인 1보다 작다고 가정한다.

표 1. 시뮬레이션 모델의 파라미터

Table 1. Parameter of Simulation model.

작업도착간격시간 (1/λ)	평균고장수리시간 (MTTR)	통신지연시간 (CDT)	평균결함발생간격시간 (MTTF)
1.5	10	0.0	2500
1.4	25	1.0	1500
1.3	250	1.5	
1.2			
1.1			

노드의 고장수리시간은 25를 기준으로 한다. 25보다 낮은 고장수리시간과 높은 고장수리시간에 관하여 시뮬레이션을 수행하여 본다. 통신지연시간은 작업처리시간의 0배, 1배, 1.5배로 상정하여 본다.[13] 작업처리시간의 0배라는 가정은 통신 지연시간이 전혀 발생되지 않음을 의미한다. 이 가정은 통신지연시간을 고려하지 않고서 재분배알고리즘의 안정성을 평가한 기존의 연구들과 비교하기 위한 것이다.

Ⅳ. 재분배 알고리즘의 효율성 평가

세가지 재분배알고리즘들의 효율성에 대한 개괄적인 이해를 위하여 재분배알고리즘의 효율성을 결정하는 요인들에 관하여 살펴볼 필요가 있다. 재분배에 관련된 효율성은 두가지 종류의 시간지연에 의해 결정된다. 첫째는 결함의 영향이 특정노드에 집중됨으로서 발생되는 비효율성이다. 만약 결함이 발생된 노드의 작업들이 정상 노드들로 분배되지 않는다면 그 작업들은 결함수리시간만큼 기다려야 한다. 비록 다른 정상노드들은 처리할 작업이 없어서 한가하더라도 특정 노드에 결함의 영향이 집중됨으로 인해서 비효율성이 발생되는 것이다. 결함집중으로 인한 비효율성은 알고리즘 A와 알고리즘 B에 서 발생되지만 알고리즘 C에서는 발생되지 않는다. 둘째는 재분배작업으로 인한 비효율성이다. 결함이 발생된 노드로부터 대기하던 작업들을 정상 노드들로 재분배시키기 위하여는 최적 노드의 탐색시간과 전송시간이 필요하며 이들 시간만큼 작업처리가 지연된다. 또한 결함노드의 작업들을 재분배하는 것이 언제나 효율적인 것도 아니다. 작업을 재분배하고 난 즉시 결함이 회복되는 경우, 결함노드의 작업재분배는 오히려 비효율적인 것이다. 이와같은 재분배 작업으로 인한 비효율성은 알고리즘 B와 알고리즘 C에서 발생되지만 알고리즘 A에서는 발생되지 않는다. 이러한 관계는 (그림 2)에 나타내었다.

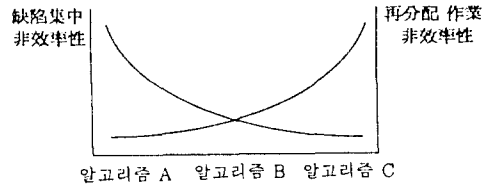

그림 2. 재분배 알고리즘 A,B,C의 상대적 비효율성

Fig. 2. Relative Inefficiency of Redistribution algorithm A,B,C.

특정 환경에서 가장 효율적인 재분배 알고리즘을 결정하기 위하여는 (그림 2)의 재분배 작업으로 인한 비효율성과 결함집중으로 인한 비효율성중 어느 것이 우세한가를 결정하여야 한다. 분산 시스템에 있어서 이러한 비효율성은 평균 대기큐의 길이(AQL:Average Queue length)의 증가로 표현된다. 그러므로 큐의 길이에 영향을 미치는 요인들을 파라미터로 하여 시뮬레이션하여 봄으로써 어떤 환경에서 어떤 재분배 알고리즘이 효율적인가를 판단할 수 있다.

1. 작업 도착간격시간의 영향

작업 도착간격시간에 따른 알고리즘의 영향을 시뮬레이션한 결과는 (그림 3)과 같다. 결함노드의 모든 작업들을 재분배시키는 알고리즘 C가 가장 우수함을 알 수 있다. 알고리즘B는 알고리즘 C보다 2배에서 3배정도의 큐길이를 나타내고 있으며 알고리즘 A는 알고리즘 C에 비해 10배에서 20배 정도의 큐길이를 보여주고 있다.

작업 도착간격시간의 변화는 각 알고리즘의 상대적인 효율성에 별다른 영향을 주지 못하는 것으로 나타났다. 다만 작업 도착간격시간이 짧아질 수록 즉 작업부하가 클수록 알고리즘 A와 알고리즘 C의 효율성 격차는 감소되는 것으로 나타났다. 이는 작업 도착간격시간이 짧을 수록 알고리즘 C에서 재분배 작업으로 인한 비효율성이 증가하기 때문인 것으로 해석할 수 있다.

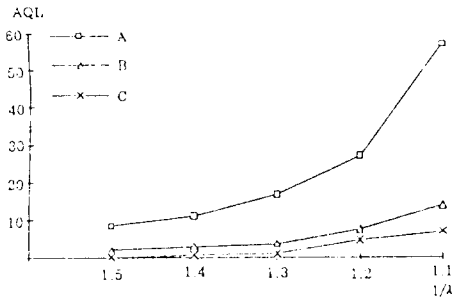

그림 3. 작업도착율이 큐의 길이에 미치는 영향 (MTTR:25, CDT:0)

Fig. 3. Influence of job arrival Rate upon queue length(MTTR:25, CDT:0).

2. 결함수리시간의 영향

결함의 종류에 따라서는 자동수리가 불가능한 경우가 있다. 자동수리가 불가능한 결함의 수리시간은 일반적으로 상당히 크다. 결함수리시간이 큰 경우에 관하여 시뮬레이션하여본 결과는 (그림 4)와 (그림 5)에 나타나 있다.

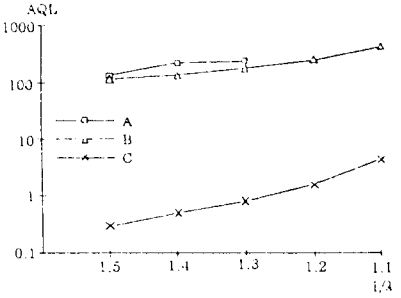

그림 4. 고장수리시간이 250일때 작업 도착율과 큐 길이의 관계(CDT:0)

Fig. 4. Relation of arrival rate to Queue length at the repair time 250(CDT: 0).

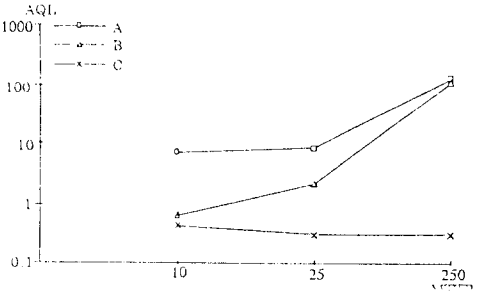

그림 5. 수리시간이 큐길이에 미치는 영향 (CDT: 0, $1/\lambda$:1.5)

Fig. 5. Influence of repair time upon Queue length(CDT:0, 1λ :1.5).

(그림 3)과 (그림 4)를 비교하여 보면 알고리즘 B와 알고리즘 C의 격차가 현저히 증가하였음을 알 수 있다. 결함수리시간이 클수록 결함집중으로 인한 비효율성이 증가된다. 결함집중으로 인한 비효율성이 발생되는 알고리즘 A와 알고리즘 B는 결함수리시간이 클수록 그 성능이 기하급수적으로 저하된다 (그림 5)로부터 통신지연이 없는 경우는 수리시간의 길이에 관계없이 알고리즘 C는 동일하며 수리시간이 증가할수록 다른 알고리즘에 비해 효율성이 더욱 우수함을 알 수 있다.

3. 통신지연시간의 영향

통신지연시간의 크기는 통신선의 속도, 위상(Topology), 노드간 거리 등의 요인들이 복합되어 영향 받을 것이다. 즉 라인 큐의 길이가 크다는 것은 이 요

인들이 내포되었음을 의미한다. 통신지연을 고려하여 시뮬레이션한 결과 (그림 3),(그림 4),(그림 5)를 통하여 알고리즘 C가 가장 효율적인 알고리즘이었으나 이는 통신지연시간을 고려하지 않았기 때문에 왜곡된 결과라 할 수 있다. 통신지연시간이 고려되지 않는 (그림 3)의 경우 재분배 작업에 따른 비효율성이 반영되지 않았으며 그 결과 알고리즘 C는 실제 이상으로 과대평가된다. (그림 6)은 통신지연시간이 1.0일 때의 알고리즘 A와 알고리즘 B및 알고리즘 C의 큐 길이를 나타낸 것이다. (그림 3)에 비해 (그림 6)의 결과는 알고리즘 A와 알고리즘 B및 알고리즘 C간의 격차가 그다지 크지 않음을 보여주고 있다. 이는 (그림 2)에서 설명되었던 재분배 작업에 의한 비효율성이 반영되었기 때문이다. 그러나 여전히 작업 도착간격시간과는 상관없이 알고리즘 C가 가장 효율적인 것으로 나타났다. 이는 작업도착간격시간이 재분배알고리즘의 효율성에 미치는 영향이 미미하다는 앞서의 결과를 확인하여 준다.

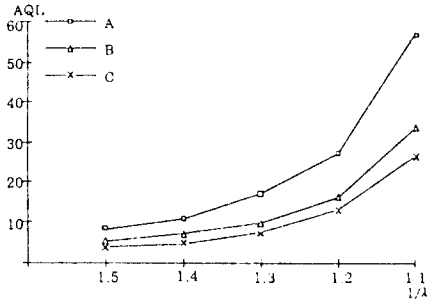

그림 6. 통신지연이 큐의 길이에 미치는 영향 (MTTR:25,CDT:1.0)

Fig. 6. Influence of CDT upon Queue Length(MTTR:25,CDT:1.0).

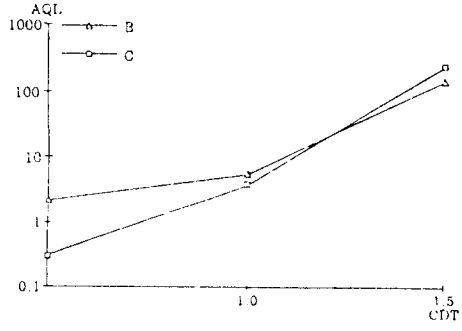

그림 7. 통신지연이 큐의 길이의 관계(MTTR:25, $1/\lambda$:1.5)

Fig. 7. Influence of CDT upon Queue Length(MTTR:25,$1/\lambda$:1.5).

그러나 알고리즘 C가 모든 상황에서 가장 효율적인 알고리즘이라고 할 수는 없다. 재분배작업에 따른 시간지연이 큰 상황에서는 알고리즘 C가 알고리즘 B나 알고리즘 A에 비해 비효율적일 가능성이 있다. 그런데 재분배 작업에 따른 시간지연에 영향을 미치는 가장 중요한 요인은 통신지연시간이다. 통신지연시간이 큰 환경(평균작업처리시간의 1.5배)에서 시뮬레이션한 결과는 (그림 8)에 보였다.

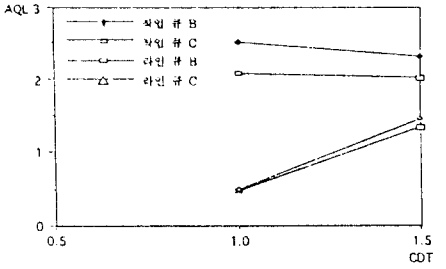

그림 8. 통신지연시간이 작업큐와 라인큐 길이에 미치는 영향(MTTR:10,$1/\lambda$:1.5)

Fig. 8. Influence of CDT upon Job Queue and Line Queue(MTTR:10,$1/\lambda$:1.5).

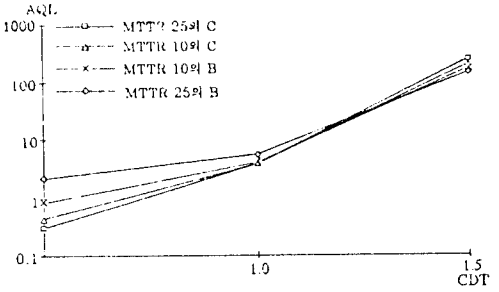

그림 9. 통신지연이 큐의 길이에 미치는 영향 ($1/\lambda$:1.5)

Fig. 9. Influence of CDT and MTTR upon Queue length($1/\lambda$:1.5).

(그림 7),(그림 9)로부터 통신지연시간이 큰 경우에는 알고리즘 C보다 알고리즘 B가 효율적임을 알 수 있다. 이는 알고리즘 C가 알고리즘 B에 비해 재분배작업을 훨씬 많이 수행하기 때문이다. 즉 알고리즘 B는 결함 발생시점에 큐내에 대기중이던 작업들만을 재분배하는데 비해 알고리즘 C는 새로이 도착하는 큐내의작업들도 재분배한다. 통신지연시간이 큰 경우에는 재분배작업으로 인한 비효율성이 결함집중으로

인한 비효율성을 훨씬 상회하기 때문에 알고리즘 C는 알고리즘 B에 비해 낮은 성능을 보인다.

통신지연이 작업 큐와 라인 큐의 길이에 미치는 영향은 (그림 8)에 나타내었다. 통신지연이 클수록 라인 큐는 알고리즘 C가 크지만 통신지연이 서비스 시간과 같거나 작으면 전체 큐의 길이는 알고리즘 B가 크며, 수리시간이 길수록 두 알고리즘간의 격차는 더욱 크리라는것은 쉽게 유추할 수있다. 이러한 결과는 알고리즘 B에 있어서 수리시간의 증가로 인하여 대기큐의 길이가 길어지기 때문이다. (그림 4)에서 살펴본 결함수리시간의 영향은 통신지연시간을 고려하지 않았기 때문에 알고리즘 C를 과대평가한 것이었다. 그러나 통신지연시간을 고려하더라도 결함수리시간은 재분배 알고리즘의 효율성에 그다지 큰 영향을 미치지 않는다고 생각할 수 있다. 왜냐하면 결함수리시간이 길어지는 경우 재분배 작업에 따른 비효율성이 증가하는 만큼 결함집중으로 인한 비효율성도 증가할 것이기 때문이다. (그림 4)와 (그림 9)를 비교하여봄으로서 이러한 추론을 확인할 수 있다. 먼저 통신지연이 큰 경우는 결함수리시간에 관계없이 알고리즘 B가 알고리즘 C보다 효율적임을 알 수 있다. 이는 부분적인 재분배로 인한 통신지연이 감소되고 결함 후 도착된 큐내의 작업들은 짧은 시간내에 복구되어 실행될 수 있기 때문이다. 또한 통신 지연이 클수록 알고리즘 C의 라인 큐의 길이가 급증하기 때문에 알고리즘 B가 훨씬 우수하다. 그러나 통신지연시간이 평균 작업처리시간과 비슷하거나 작은 경우는 수리시간에 관계없이 알고리즘 C가 우수함을 알수있다.

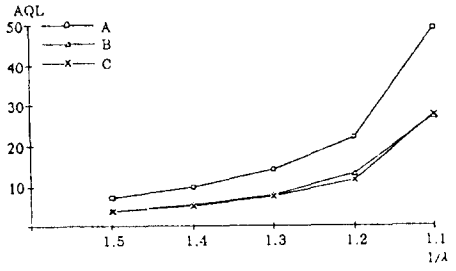

그림 10. 통신지연이 큐의 길이의 관계 (MTTR:10, CDT:1/0)

Fig. 10. RELATION OF CDT to Queue length (MTTR:10, CDT:1/0).

결함수리시간이 10이라고 가정하고 시뮬레이션한 결과는 (그림 10)에 제시되었다. (그림 6)과 (그림 10)을 비교할때 비록 알고리즘 B와 알고리즘 C의 격차가 감소되긴 하였지만 (그림 10)의 결과는 (그림 6)의 결과와 거의 유사하다는 점을 알 수 있다. 여전히 알고리즘 C가 알고리즘 B보다 효율적인 것으로 나타났다. 이는 결함수리시간이 알고리즘의 효율성에 그리 커다란 영향을 주지 못함을 의미한다.

4. 결함발생간격시간의 영향

과연 고장이 잦은 시스템과 고장이 드문 시스템들간에 있어서 상이한 재분배 알고리즘을 사용해야 하는가는 결함발생간격이 재분배 알고리즘의 효율성에 얼마나 영향을 주느냐의 문제와 동일하다. (그림 11)은 통신지연이 없고 평균 수리시간이 10일경우 결함발생간격시간이 1500과 2500인 경우 시뮬레이션한 결과를 도시한 것이다. 결함발생간격시간의 변화가 대기행렬의 길이에 거의 영향을 미치지 못함을 볼 수 있다. 즉 결함발생간격시간이 부하 재분배알고리즘들의 효율성에 별다른 영향을 미치지 않음을 의미한다. 재분배 알고리즘의 선택에 있어서 고장의 빈도는 그다지 중요하지 않음을 알 수 있다.

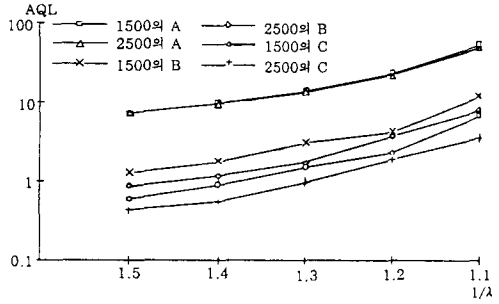

그림 11. 결함간격시간이 큐길이에 미치는 영향 (MTTR:10, CDT:0)

Fig. 11. Influence of MTTF upon Queue length (MTTR:10, CDT:1/0).

Ⅴ. 시뮬레이션 결과의 시사점 및 결론

시뮬레이션 결과 작업의 도착간격시간.결함발생간격시간 및 결함수리시간은 재분배 알고리즘들간의 상대적 효율성에 큰 영향을 주지 않음을 보았다. 통신지연시간의 변화는 재분배알고리즘의 상대적 효율성에 커다란 영향을 준다. 통신지연시간이 평균 작업처리시간 보다 큰 경우에는 알고리즘 C보다는 알고리즘 B가 우수하며 통신지연시간이 평균작업처리시간과

유사하거나 작을 경우에는 알고리즘 B나 알고리즘 A보다는 알고리즘 C가 현격하게 우수하다 본 논문에서는 결함허용 분산시스템의 부하 재분배 알고리즘의 성능을 평가하기 위한 시뮬레이션 모델을 작성한 후 시뮬레이션을 통하여 다양한 환경속에서 어떤 재분배 알고리즘들이 바람직한지를 분석해 보았다. 본 논문에서 제시된 분석결과가 일반화되기 위하여는 보다 대규모의 분산시스템에 관한 연구가 수행되어야 할 것이다. 본 논문에서는 시뮬레이션을 통하여 재분배 알고리즘들의 성능을 비교적 구체적으로 평가할 수 있다는 점을 확인할 수 있었다. 본 논문에서 고려하지 못한 한계점과 향후 연구과제는 다음과 같다.

① 시뮬레이션 프로그램의 용량한계로 인한 대규모 분산시스템에 관한 시뮬 레이션이 어려웠다.

② 분산 시스템의 각 노드간 거리 또는 통신속도를 고려하지 않았다.

③ 부하지표가 프로세스단위인 경우 프로세스간의 통신지연 파라미터를 고려한 재분배 알고리즘의 연구가 필요하다.

부 록

```
GEN, BKCHOI, FAULT TOLERANT SYSTEM C,5/30/93,1;
LIMITS,30,2,1300;
INTLC, XX(1)=1,XX(2)=1,XX(3)=1,XX(4)=1,XX(5)=1;
NETWORK;
      RESOURCE/PPA1(1),1;
      RESOURCE/PPB2(1),2;
      RESOURCE/PPC3(1),3;
      RESOURCE/PPD4(1),4;
      RESOURCE/PPE5(1),5;
      RESOURCE/DLA1(1),6;
      RESOURCE/DLB2(1),7;
      RESOURCE/DLC3(1),8;
      RESOURCE/DLD4(1),9;
      RESOURCE/DLE5(1),10;
      CREATE, EXPON(1.50);
GOA   ASSIGN, ATRIB(1)=1;
      ACTIVITY,,BNEW;
      CREATE, EXPON(1.50);
GOB   ASSIGN, ATRIB(1)=2;
      ACTIVITY,,BNEW;
      CREATE, EXPON(1.50);
GOC   ASSIGN, ATRIB(1)=3;
      ACTIVITY,,BNEW;
      CREATE, EXPON(1.50);
GOD   ASSIGN, ATRIB(1)=4;
      ACTIVITY,,BNEW;
      CREATE, EXPON(1.50);
GOE   ASSIGN, ATRIB(1)=5;
      ACTIVITY,,BNEW;
      CREATE, EXPON(1.50);
BNEW  ASSIGN, II=ATRIB(1);
      ASSIGN, XX(30)=XX(1)*NNQ(1)+XX(2)*NNQ(2)+XX(3)*NNQ(3);
      ASSIGN, XX(30)=XX(30)+XX(4)*NNQ(4)+XX(5)*NNQ(5);
      ASSIGN, XX(31)=XX(1)+XX(2)+XX(3)+XX(4)+XX(5);
      ASSIGN, XX(30)=XX(30)/XX(31);
      GOON, 1;
         ACT,,XX(II).EQ. 0, LP;
         ACT,,XX(30) .LT. NNQ(II), LP;
         ACT,,XX(II) .EQ. 1, WAIT;
LP    ASSIGN, XX(32) = XX(1)+XX(2)+XX(3)+XX(4)+XX(5);
      ASSIGN, II=ATRIB(1)+5, ATRIB(1)=II;
      AWAIT(ATRIB(1)=6,10),ATRIB(1)/1;
      ACT,1.5;
      FREE,ATRIB(1)/1;
      GOON,1;
      ACT,,XX(32).EQ. 0,DETH;
XINIT ASSIGN, XX(50)=1999,XX(51)=0,II=0;
ADDII ASSIGN, II=II+1;
      GOON,1;
        ACT,,II      .GT. 5, ROUTE;
        ACT,,XX(II) .EQ. 0,  ADDII;
SLECT GOON,1;
        ACT,,XX(50) .LE. NNQ(II), ADDII;
        ACT,,XX(50) .GT. NNQ(II), XCHNG;
XCHNG ASSIGN,XX(50)=NNQ(II),XX(51)=II;
      ACT,,ADDII;
ROUTE GOON,1;
      ACT,,XX(51) .EQ. 1, GOA;
      ACT,,XX(51) .EQ. 2, GOB;
      ACT,,XX(51) .EQ. 3, GOC;
      ACT,,XX(51) .EQ. 4, GOD;
      ACT,,XX(51) .EQ. 5, GOE;
WAIT  AWAIT(ATRIB(1)=1,5), ATRIB(1)/1;
      ASSIGN, II=ATRIB(1);
      GOON,1;
         ACTIVITY,,XX(II) .EQ. 0, DSTBR;
         ACTIVITY,,XX(II) .NE. 0, NEXT;
NEXT  GOON,1;
      ACT,,ATRIB(1).EQ. 1, PRA;
      ACT,,ATRIB(1).EQ. 2, PRB;
      ACT,,ATRIB(1).EQ. 3, PRC;
      ACT,,ATRIB(1).EQ. 4, PRD;
      ACT,,ATRIB(1).EQ. 5, PRE;
PRA   GOON,1;
      ACT,EXPON(1.0),,FRR;
PRB   GOON,1;
      ACT,EXPON(1.0),,FRR;
PRC   GOON,1;
      ACT,EXPON(1.0),,FRR;
PRD   GOON,1;
      ACT,EXPON(1.0),,FRR;
PRE   GOON,1;
      ACT,EXPON(1.0),,FRR;
FRR   FREE,ATRIB(1)/1;
DETH  TERMINATE;
DSTBR FREE, ATRIB(1)/1;
      ACT,,LP;
      CREATE;
      ACT,EXPON(2500.);
AFALT ASSIGN, XX(1) = 0, ATRIB(1)=NNQ(1), ATRIB(2)=-1*ATRIB(1);
      PREMPT(11),PPA1,LP,2;
      ALTER, PPA1/ATRIB(1);
        ACT,,0001;
      ALTER, PPA1/ATRIB(2);
        ACT,10;
      FREE, PPA1/1;
      ASSIGN, XX(1)=1;
      ACT,EXPON(2500.),,AFALT;
      CREATE;
      ACT,EXPON(2500.);
BFALT ASSIGN, XX(2) = 0, ATRIB(1)=NNQ(2), ATRIB(2)=-1*ATRIB(1);
      PREMPT(12),PPB2,LP,2;
      ALTER, PPB2/ATRIB(1);
      ACT,,0001;
      ALTER, PPB2/ATRIB(2);
        ACT,10;
      FREE, PPB2/1;
      ASSIGN, XX(2)=1;
      ACT,EXPON(2500.),,BFALT;
      CREATE;
      ACT,EXPON(2500.);
CFALT ASSIGN, XX(3) = 0, ATRIB(1)=NNQ(3), ATRIB(2)=-1*ATRIB(1);
      PREMPT(13),PPC3,LP,2;
      ALTER, PPC3/ATRIB(1);
      ACT,,0001;
      ALTER, PPC3/ATRIB(2);
        ACT,10;
      FREE, PPC3/1;
      ASSIGN, XX(3)=1;
      ACT,EXPON(2500.),,CFALT;
      CREATE;
      ACT,EXPON(2500.);
DFALT ASSIGN, XX(4) = 0, ATRIB(1)=NNQ(4), ATRIB(2)=-1*ATRIB(1);
      PREMPT(14),PPD4,LP,2;
      ALTER, PPD4/ATRIB(1);
      ACT,,0001;
      ALTER, PPD4/ATRIB(2);
        ACT,10;
      FREE, PPD4/1;
      ASSIGN, XX(4)=1;
      ACT,EXPON(2500.),,DFALT;
      CREATE;
      ACT,EXPON(2500.);
EFALT ASSIGN, XX(5) = 0, ATRIB(1)=NNQ(5), ATRIB(2)=-1*AT
RIB(1);
      PREMPT(15),PPE5,LP,2;
      ALTER, PPE5/ATRIB(1);
      ACT,,0001;
      ALTER, PPE5/ATRIB(2);
        ACT,10;
      FREE, PPE5/1;
      ASSIGN, XX(5)=1;
      ACT,EXPON(2500.),,EFALT;
      END;
INIT,0,100000;
FIN;
```

參 考 文 獻

[1] T.C.K.Chou, J.A.Abraham, "Load Redistribution Under Failure in Distributed Systems", *IEEE Trans. on Comp.* Vol. C-32, No.9, PP.799-808, Sept.1983.

[2] J.S.Park, "A Stability Criterion and Load Redistribution in a Fault Tolerant System", Ph.D.Thesis. Chung Ang Univ., Dec.1990.

[3] A.M.Law & W.D. Kelton, Simulation Modeling and Analysis, McGraw-Hill Inc., 1991.

[4] Y.C. Chow, W.H. Kohler, "Models for Dynamic Load Balancing in a Heterogeneous Multiple Processor Systems", IEEE Trans.on Comp., Vol c-28, No.5, PP.354-361, May, 1979.

[5] B.T. Dosh, H. Heffes, "Overload Performance of several Processing Queuing Disciplines for the M/M/1 Queue", IEEE trans. on Comm., Vol.C-34, No.6, june 1986.

[6] A.L. Hopkins, T.B. Smith III, J.H. Lala, "FTMP-A Highly Reliable Fault Tolerant Multiprocessing for Aircraft", Proc.of the IEEE, PP.1240-1255, Oct. 1978.

[7] W.N. Toy, "Fault-Tolerant Design of Local ESS processors", Proc.of the IEEE Vol.66, No.10, PP.1126-1145, Oct.1978.

[8] J.H. Wensley, L. Lamport, J. Goldberg, M. W. Green, K. N. Levitt, P. M. Mellar Smith, R.E. E. Shostak, C.B. Weinstock, "SIFT: Design and Analysis of a Fault-Tolerant Computer for Aircraft Control", Proc. of the IEEE Vol.66, No. 10, PP.1240-1255, Oct.1978.

[9] J.M. Ayache, J.P. Courtiat, M. Diaz, "REBUS, A Fault-Tolerant Distributed System for Industrial Real-Time Control", IEEE Trans. on Computer, Vol.c-31, no.7, PP.637-647, July 1982.

[10] D.P. Siewiorek, "Fault Tolerance in Commercial Computers", IEEE Computer Vol.23, No.7, PP.26-37, July, 1990.

[11] V.P. Nelson and B.D. Carroll, "Tutorial: Fault Tolerant Computing", IEEE Catalog No.EH0254-3, ISBN0-8186-0677-0, 1987.

[12] A. Alan, B. Pritsker, "Introduction to Simulation and SLAM II", 3rd Ed. Systems Publishing Co., 1986.

[13] C.Y Huang, J.W.-S. Liu, "Dynamic Load Balancing Algorithms in Homogeneous Distributed Systems", Proc. 16Th Int. Symp. on Fault-tolerant Computing FTCS-16, PP.216-223, 1986.

著者紹介

崔 柄 甲(正會員)

1948年 9月 3日生. 1976年 한양대학교 전자공학과 졸업. 1982年 동대학원졸업. 1990年 청주대학교 박사과정수료. 1976年 삼미종합특수강(주)전산실. 1979年 한국원자력연구소전산실 선임연구원. 1984年 광주대전산과 전임강사. 1986年 ~ 현재 목원대 컴퓨터공학과 부교수. 주관심 분야는 시스템 소프트웨어, 데이터베이스, 소프트웨어공학 등임.

李 天 熙(正會員)

1945年 6月 6日生. 1968年 한양대학교 전자공학과 졸업 동대학원 졸업. 1975年 성균관대학교 대학원 전자자료처리학과 졸업. 1986年 성균관대학교 전자공학과 공학박사학위 취득. 1971年 (주)한국마벨 근무. 1977年 동양공업전문대학 전자공학과 근무. 1979年 ~ 현재 청주대학교 전자공학과 교수. 1983年 ~ 1985年 미국 캘리포니아 산호세 주립대학교 전산과 객원교수. 주관심 분야는 VLSI Layout, ASIC, DRAM, CAD TOOL개발 등임.