소프트웨어 재활 기법을 적용한 다중계 시스템의 가용도 분석

(Availability Analysis of Multiplex Systems using Software Rejuvenation Method)

박 기 진 † 김 성 수 †† 김 재 훈 †††
(Kiejin Park) (Sungsoo Kim) (Jai-hoon Kim)

요 약 고가용성 다중계 시스템의 소프트웨어 재활 기법은 시스템의 결함 발생 이후에 수동적으로 대처하기보다는 결함이 발생하기 전에 이를 미연에 방지하는 능동적 차원의 결함 허용 방법이다. 특히 멀티미디어 이동 컴퓨팅에서 사용되는 소프트웨어는 통신 단절, 데이터 유실 등으로 인한 노화 진행이 일반 소프트웨어보다 상당히 빠르게 진행되기 때문에 소프트웨어 재활에 의한 결함 예방 방법은 대규모 멀티미디어 이동 컴퓨팅 시스템에 사용될 가능성이 높다고 볼 수 있다. 본 연구에서는 서버에서 수행되는 소프트웨어의 재활 주기, 재활 소요시간, 서버의 고장률, 수리률, 동시에 가동되는 서버의 수, 서버의 가동 기간 및 가동 방식 등의 시스템 운영 파라미터에 기초하여, 소프트웨어 재활 정책에 대한 평가를 위한 평형 상태에서의 확률, downtime, 가용도, 손실 비용 등을 계산하였다. 수학적 분석을 통해 구한 재활 모델의 closed-form 해는 다양한 시스템 운영 상태에 대한 실험을 통해 검증하였으며, 소프트웨어 재활을 통한 예방적 결함허용 기법의 적용 가능성이 높다는 것을 확인하였다. 또한 서버의 고장률 및 불안정률이 소프트웨어 재활 정책 결정에 중요한 요소임을 파악하였다.

Abstract The software rejuvenation method for highly available multiplex systems uses a pro-active fault-tolerant approach to handle system failures. The software rejuvenation prevents failures from occurring, while the previous methods recover from failures after happening. Especially, since the software aging proceeds fast in the software used for the multimedia mobile computing due to the loss of communications or data, the preventive method from failures using software rejuvenation can be used for the multimedia mobile computing. In this paper, according to the operational parameters such as rejuvenation period, rejuvenation time, failure rate and repair rate of the servers, number of running servers, duration of running time, and type of running modes, we calculate steady-state probabilities, downtime, availability, and cost of the multiplex systems using software rejuvenation method. We validate the closed-form solutions of the mathematical model by experiments based on various operational parameters and find that the software rejuvenation method can be adopted as preventive fault-tolerant technique. The failure rate and unstable rate of the servers are essential factors for the decision making of the rejuvenation policies.

· 본 연구는 2000년도 교육부 Brain Korea 21(BK21) 지원 사업에 의한 결과임.

† 비 회 원 : 아주대학교 컴퓨터공학과
kiejin@yahoo.co.kr
†† 종신회원 : 아주대학교 정보통신전문대학원 교수
sskim@madang.ajou.ac.kr
††† 정 회 원 : 아주대학교 정보통신전문대학원 교수
jaikim@madang.ajou.ac.kr

논문접수 : 1999년 8월 28일
심사완료 : 2000년 5월 27일

1. 서 론

인터넷 · 이동통신 · 네트워크 기술 등의 발전과 더불어 서버(멀티미디어 서버, 인터넷 서버, 교환기)와 휴대단말(PDA, HPC, smart phone)로 구성된 컴퓨터 시스템은 점차 대규모화 · 복잡화되는 추세이다. 향후 통신 서비스는 2Mbps 이상의 대역폭을 이용하는 멀티미디어 및 이동컴퓨팅 환경으로 발전될 것이 확실하며, 기존 아날로그 기반의 data over voice에서 디지털 기반의

voice over data 통신 구조로 급속히 전환되고 있다[1, 2]. 인터넷과 이동 컴퓨팅 환경에서는 다양한 응용 분야가 새로이 등장하고, 컴퓨터와 통신 기술을 상호 통합(computer telephony integration: CTI)시키는 응용 프로그램의 대폭적인 증가가 예상되며, 이러한 수요에 대응하는 가용도(availability) 요구 수준이 매우 엄격한 고속, 대용량 시스템의 보급 확산이 빠르게 이루어지고 있다[3]. 최종 사용자의 입장에서 가장 중요시되는 시스템 평가 척도는 "사용자가 원하는 경우에 시스템에 접속해서 서비스를 받을 수 있는 지"를 판단하는 가용도이며, 이는 평균고장시간(mean time to failure: MTTF) 만을 다루는 신뢰도와는 달리 평균수리시간(mean time to repair: MTTR) 까지 포함하는, 즉 시스템이 고장났을 경우에 이를 얼마나 신속히 복구할 수 있는 능력을 고려한 평가 개념이다[4].

또한 컴퓨터 시스템 기술의 발전과 사용자의 다양한 요구로 인하여 소프트웨어의 복잡도는 날로 증가되는 추세에 있다. 소프트웨어의 복잡도가 증가됨에 따라 소프트웨어의 설계, 구현, 또는 그 밖의 여러 가지 원인과 소프트웨어와 관련된 결함으로 인하여 전체 시스템의 오동작 또는 수행 중단으로 이어지는 사례가 늘어나고 있다. 현재 정보화 사회의 특성상 대다수의 업무들이 컴퓨터 시스템에 대한 의존도가 절대적이고, 많은 업무에서 중단 없는 서비스가 요구되므로 컴퓨터 시스템의 장애로 인한 피해는 이루 상상할 수 없을 정도로 크다[5, 6]. 물론 컴퓨터 장애의 원인으로는 하드웨어 결함에 의한 장애도 있지만 조사된 연구에 따르면 소프트웨어에 의한 장애가 전체의 62%를 차지하는 것으로 보고되었고[7] 앞으로 소프트웨어의 결함에 의한 컴퓨터의 장애는 그 비중이 점차로 늘어날 전망이다. 특히 멀티미디어 이동 컴퓨팅에서 사용되는 소프트웨어는 빈번한 통신 두절과 데이타 유실로 인하여 소프트웨어의 결함이 더욱 심각할 가능성이 높다. .

소프트웨어의 결함을 줄이기 위한 방법으로, 소프트웨어의 결함을 설계, 구현, 시험 과정에서 발견하여 이를 해결하는 방법이 가장 이상적이지만 현실적으로 결함이 전무한 상태로 만드는 것은 불가능하다. 발견 안된 결함으로 또는 그 밖의 원인으로 인한 프로그램 수행도중의 결함을 복구하기 위하여 (1) 프로그램 수행 중 (주기적으로) 중간에 태스크 상태를 저장하여 소프트웨어 결함 시 가장 최근의 태스크 저장 상태로부터 다시 시작하는 방법으로 가용성을 높일 수 있으나 결함 시점을 예측하기 힘들고 복구되어 다시 시작될 때까지의 불시의 업무 중단으로 인한 심각한 피해를 막을 수 없고, (2) 프로그램 또는 데이타를 이중계로 구성하여 태스크를 수행하는 방법은 자원의 낭비가 심하고 복사된 프로그램 또는 데이타들의 일치성을 유지시키는데 많은 어려움이 있다[8]. 이러한 기존의 연구들은 장애가 발생했을 때 이를 발견하고 해결하는 사후처리 방식을 기본으로 한다. 이러한 사후 해결책을 기본으로 하는 방식은 불시에 장애가 발생하므로 예측이 불가능하고 장애 발생 후 복구될 때까지 컴퓨터 시스템이 서비스를 하지 못하므로 중요한 애플리케이션 응용(금융업, 통신, 원자력 이용, 항공산업 등) 분야에서 막대한 손실과 혼란을 초래할 수 있다.

본 연구에서는 멀티미디어 이동 컴퓨팅 서버의 가용도를 개선하기 위하여 별도의 하드웨어 자원을 필요로 하지 않고, 불시의 소프트웨어 장애로 인한 피해가 없이, 컴퓨터 시스템 관리자가 예측할 수 있는 적절한 시점(예를 들면, 시스템의 이용이 한산하다든지, 프로그램이 문제를 야기 시킬 우려가 있다든지)을 택하여 프로그램을 일시적으로 중지시킨 후 다시 가동시키는 소프트웨어 재활(rejuvenation) 기법을 사용하였다. 멀티미디어 이동 컴퓨팅을 위한 소프트웨어 재활 기법은 컴퓨터 시스템의 가용도를 개선하기 위해 하드웨어 혹은 소프트웨어의 결함 발생 이후에 수동적으로 대처하는 기존의 복구 방식에서 진일보한 개념으로, 결함이 발생하기 전에 이를 미연에 방지하는 능동적 차원의 예방적 유지 보수 개념의 결함 허용 방법이다. 특히 멀티미디어 이동 컴퓨팅에서 사용되는 소프트웨어는 통신 단절, 데이타 유실 등으로 인한 노화 진행이 일반 소프트웨어보다 상당히 빠르게 진행되기 때문에 소프트웨어 재활에 의한 결함 예방 방법은 대규모 멀티미디어 이동 컴퓨팅 시스템에 사용될 가능성이 높다고 볼 수 있다. 소프트웨어 재활 기법은 점차로 복잡해져 가는 인터넷, 실시간 처리 등의 클라이언트-서버 컴퓨팅 환경에 적합한 대안이라 평가되고 있고, 버퍼 플러싱, 메모리 청소, 파일시스템 정리, 커널 테이블의 초기화 등을 재시동 방법으로 처리할 수 있으며, 이 경우 소프트웨어는 일시적 결함이 발생할 확률이 작은 새로운 상태에서 재출발할 수 있게 된다. 멀티미디어 이동 컴퓨팅 서버의 가용도를 높이기 위해서, 시스템의 가동을 주기적으로 멈추자는 소프트웨어 재활 아이디어는 "시스템이 어느 시점 혹은 어떤 상태에 있을 때 정지시키는가?"에 대한 소프트웨어 재활 주기 및 조건 결정이 문제의 핵심이며, 재활 정책의 효율성은 평형 상태에서의 확률, 시스템의 downtime, 가용도 및 손실 비용 등의 척도로써 측정될 수 있다.

2. 관련 연구

멀티미디어 이동 서비스를 위한 컴퓨터 시스템에서 소프트웨어 노화로 인한 결함(결함허용 분야에서 이와 같은 유형의 결함을 "heisenbugs"라고 함)의 감지, 수정은 상당히 어려우며, 단순히 하드웨어적으로 시스템의 가용도를 높이기보다는 소프트웨어 결함으로 인한 시스템 장애를 사전에 예방함으로써, 서비스가 거부되는 평균 횟수를 최소화하는 것이 오히려 더 바람직하다. 하드웨어 기술의 발전으로 인하여 시스템 가용도에 하드웨어보다는 소프트웨어가 더욱더 큰 영향을 끼치고 있으며, 특히 복잡한 대규모 소프트웨어의 등장으로 결함이 없는(fault-free) 소프트웨어의 개발은 거의 불가능하게 되므로 인해, 소프트웨어 결함허용에 대한 필요성이 점차 중요시되고 있다. 소프트웨어 결함허용을 위해서는 정성적 방법이 아닌 정량적 방법을 적용한 평가 기법이 요구되고 있으며, 현재 사용되고 있는 방법은 다음과 같다.

- récovery block : 오류가 발생하면 이전 시점으로 되돌아가 같은 기능을 가진 다른 모듈을 실행하는 방법
- N-version programming : N개의 독립적인 소프트웨어 모듈이 수행한 결과를 비교하여, 다수의 동일한 결과를 채택하는 방법
- N-self checking programming : 수행 모듈에 결함이 발생하면, 대기 중인 모듈중의 하나가 새로운 수행 모듈이 되어 주어진 기능을 수행하는 방법
- checkpointing : 결함이 발생할 경우, 작업의 완료 시간을 최소화하는 기법으로써, 결함이 발생하면 가장 최근에 저장된 프로세스 상태(checkpoint)를 가지고 작업을 재 수행하는 방법

기존의 소프트웨어 결함허용 방식은 앞서 언급한 새로운 환경에 대한 적응력이 떨어지며, 네트워크 환경과 응용 프로그램의 복잡·대형화 추세로 인해, 기존에 사용된 기법만으로 요구되는 소프트웨어 의존도(dependability)를 제공하기란 거의 불가능하므로, 대규모 실시간 트랜잭션 처리 응용 프로그램(인터넷 검색 서비스, 뱅킹, 증권 전산 등)들의 가용도 모델링에는 적합하지 않다. 클라이언트-서버 형태의 시스템에서 서버 소프트웨어는 상당히 긴 시간동안 수행 가능하여야 하며, 서버 소프트웨어의 가동 시간이 길어질수록(*software aging*) 다수의 클라이언트의 요청으로 인한 에러 데이타의 누적이 필연적이다[9]. 소프트웨어 노화로 인해서 시스템의 성능 저하 및 일시적 결함의 발생 가능성이 커지게 되며, 결국에는 메모리 부족, 화일 공유 오류, 또는 데이타 손상 등과 같은 소프트한 결함으로 인해서, 서버 소프트웨어의 성능이 점차로 저하되어, 이것이 지속될 경우 시스템 작동이 멈추게 된다. 일반적으로 시스템의 성능이 최고 수준에 도달했을 경우, 재활 작업을 시작해야 하는 시점으로 판단하고 있으며, 소프트웨어를 주기적으로 재가동시킴으로서, 소프트웨어 결함으로 인한 시스템 다운 발생 확률을 감소시킬 수 있다.

Trivedi[10,11,12,13]는 버퍼 용량과 서비스 부하를 고려한 소프트웨어 재활 모델을 제시하였으며, 최적 재활 주기 및 작업 손실 확률(loss probability) 등을 구하였으나, 유지보수 정책 평가를 위한 비용 함수에 대한 고려가 부족하다. [14]의 경우, 소프트웨어 재활 아이디어를 처음으로 제시한 논문으로 서버가 머물 수 있는 상태를 마르코프 체인으로 모델링 하여, 수학적 해를 구하는 기초를 제공했으며 모든 상태에 머무는 시간은 지수분포를 따른다고 가정하고 있다. Trivedi의 경우 이 가정을 완화시켜, 어느 특정 상태에서 머무는 시간이 memoryless property를 가지지 않는 semi-markov process에 관한 것도 다루었으며, 또한 Petri Net 모델링 기법을 이용하여 소프트웨어 재활 기법의 적용 가능성을 연구한 사례도 찾을 수 있다[11]. Huang[15, 16, 17]은 유지 보수 기간동안에 서버가 가동하지 않아서 발생하는 비용 증가를 고려한 최적 원상 복구 작업 조건을 구하였으나, state transition model 이 단순하며, 서버가 한 대만 가동되는 simplex 시스템만을 다루었다. 또한 작업 완료 시간을 최소화하기 위해, checkpointing [18] 기법과 재활 기법을 결합하여 연구한 사례도 있으며, 최근 Motorola[19]에서는 무선기기의 고장 및 에러를 관리하기 위해, 기존의 유선 라인에서 사용되었던 하드웨어 이중화 기법 대신에, 소프트웨어 기법을 활용한 이동 컴퓨팅 환경의 서버와 정보 단말기간의 동적 데이타에 대한 재활 전략을 연구 중에 있다.

멀티미디어 이동 컴퓨팅의 특성상 무정지 요건을 갖추기 위해서는 고수준의 가용성을 보장해 주는 다중계 시스템 구성이 일반적이지만(예를 들어 이중계로 구성된 시스템에서는 두 대의 서버가 동시에 가동되며, 만약 어느 한 서버에 이상이 발생할 경우, 즉각 다른 서버에서 업무를 처리하게 된다.) 지금까지의 연구 결과는 단 한 대의 서버에서 가동되는 소프트웨어 재활 기법에 관한 분석이 주를 이루고 있다. 본 논문에서는 교환기, 대용량 트랜잭션 서버와 같이 이중계 이상의 서버에서 가동되는 소프트웨어 시스템의 가용도를 계산하기 위해, n개

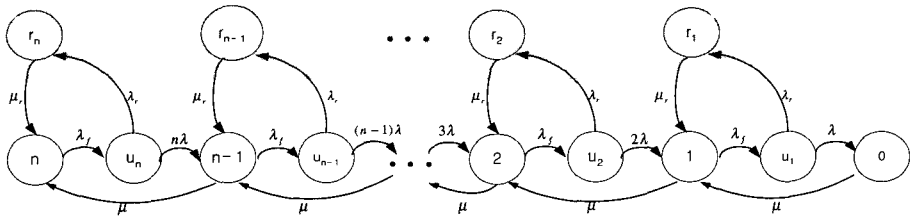


그림 1 소프트웨어 재활을 고려한 HSS 시스템 상태 모델

의 서버가 동시에 가동되는 hot standby sparing(HSS) 시스템과 n개의 서버가 있으나 특정 시간에 가동되는 서버는 오직 한 개이며, 이것이 고장 났을 경우에만 여분 서버 중 1개가 가동에 들어가는 cold standby sparing(CSS) 시스템[8]에서의 소프트웨어 재활에 관한 주제를 다루었다.

본 연구에서는 기존에 기계 장비의 유지보수에 사용되던 예방적 유지보수(preventive maintenance)기법에 기초한 소프트웨어 재활 기법을 활용하여, 시스템 정지로 인하여 발생되는 비용이 최소가 되도록 하는 소프트웨어 재활 정책을 수학적 방법으로 계산하였으며, 재활 정책 평가를 위한 비용 함수를 정의하여 최적 재활 주기를 구하였다. 또한 서버의 가동 방식(HSS, CSS)에 따른 다양한 성능 파라미터를 계산하였다.

본 논문의 2장에서는 관련 연구 결과와 문제를 정의하였으며, 3장에서는 소프트웨어 재활을 적용한 서버 가동에 대한 시스템 모델을 정의하고, 4장에서는 제안된 모델의 수학적 해석 방법의 정확성을 검증하기 위한 다양한 시스템 운영 조건에 대한 실험을 수행하였다. 마지막으로 결론부에서는 제안된 소프트웨어 재활 기법의 활용 방안 및 향후 연구에 관하여 논하였다.

3. 시스템 모델

대부분의 소프트웨어 결함이 영구적이기보다는 일시적 결함(transient fault)의 성질을 가지고 있으며, 소프트웨어 노화로 인해 발생되는 일시적 결함은 시스템을 재 가동할 경우에 대부분 없어진다. 서버 가동 방식(HSS 혹은 CSS)에 따른 결함 발생 모델 분석을 통해 획득한 해석적 수치에 근거하여 다양한 결함 허용 기준을 평가할 수 있다. 그림 1은 소프트웨어 재활을 고려한 HSS 시스템의 운영 상태 모델을 나타내며, 시스템 운영 모델링에 사용된 가정들은 다음과 같다.

- n개의 서버로 구성된 시스템에서 각 서버의 고장률 (λ)은 동일하다.
- 고장 난 서버를 수리하는 수리률(μ)은 동일하다.
- HSS 시스템에서 재활에 들어갈 경우, 주서버를 제외한 나머지 여분 서버가 재활 작업 대상이며, 재활 작업 시간($1/\mu_r$)은 여분 서버 수에 무관하다.
- CSS 시스템에서 재활에 들어갈 경우, 현재 가동하고 있는 주서버가 재활 작업 대상이며, 가동되고 있지 않는 서버는 고려 대상에서 제외된다. 재활 작업 시간$1/\mu_r$)은 서버 수에 무관하다.
- 서버의 가동을 주기적으로 멈추는 재활률(λ_r)은 모든 가동 상태에서 동일하다.
- HSS 시스템에서 다른 서버로의 작업 전이 시간은 극히 짧으며, 이 시간은 무시할 수 있다.
- HSS 시스템에서는 재활이 완료된 서버 중 하나가 기존에 가동중인 서버를 대치한다. (즉, 현재 제공되는 서비스의 중단 없이 재활 작업을 수행한다.)
- 모든 state에 머무는 시간은 지수분포를 따른다. (즉, memoryless property를 가짐)

그림 1의 시스템 상태 모델은 위와 같은 가정 하에서 irreducible recurrent nonnull markov chain을 형성하므로, 평형 상태에서 확률 값을 비교적 쉽게 구할 수 있게 된다. 만약 위의 state 중 어느 한가지라도 memoryless property를 만족하지 않는다면 semi-markov process로 모델링이 되어, 정확한 해를 구하는 간단한 방법이 존재하지 않는다[20,21].

정상 상태(normal state)에서 가동되고 있는 서버는 n, n-1, ..., 1, 0 등의 가동 중인 서버의 수를 state로 가지고 있으며, 장시간 가동으로 인해 성능이 저하된 상태(unstable state)의 서버는 u_n, u_{n-1}, ..., u_2, u_1으로 나타냈다. unstable state에서는 λ_r의 변화율로 재활 작업에 들어가거나, 혹은 $i*\lambda$의 변화율(HSS 시스템의 경우 고장이 나지 않은 i 대의 서버가 동시에 가동함)로 고장이 발생하게 된다. 또한 normal state에서 unstable state로의 변화율은 λ_f로 표시되며, 이는 소프트웨어의 장기간 가동으로 인한 시스템의 불안정성을 반영한다.

회색으로 칠해진 영역의 r_n, r_{n-1}, ..., r_2, r_1은 재활 상태(rejuvenation state)를 표시하며, 시스템의 가동을 고의로 중지시켜 재 부팅하는 state를 나타낸다.

그림 1의 평형 상태 (임의의 state에서 입력과 출력량이 같아짐)에서의 balance equation은 다음과 같다. (여기서 P_i는 임의의 시간에 시스템이 상태 i 에 머물 확률을 의미함)

$$\lambda_f P_n = \mu_r P_{r_n} + \mu P_{n-1}$$

$$(\lambda_f + \mu) P_{n-i} = (n-i+1)\lambda P_{u_{n-i+1}} + \mu_r P_{r_{n-i}} + \mu P_{n-i-1}, \text{i= } 1,2,...,n-1$$

$$(\lambda_r + (n-i)\lambda) P_{u_{n-i}} = \lambda_f P_{n-i}, \text{ i = } 0,1,...,n-1$$

$$\mu_r P_{r_{n-i}} = \lambda_r P_{u_{n-i}}, \text{ i = } 0,1,...,n-1$$

$$\mu P_0 = \lambda P_{u_1}$$

위의 balance equation과 각 state에서 머물 확률의 총합이 1이 되는 다음 식을 결합한 연립 방정식을 풀면, 평형상태에서의 각 state 머물 확률(steady-state probability)을 얻을 수 있다.

$$\sum_{i=0}^{n} P_i + \sum_{i=1}^{n} (P_{u_i} + P_{r_i}) = 1$$

$$P_{u_{n-i}} = \frac{\lambda_f}{\lambda_r + (n-i)\lambda} P_{n-i}, \text{ i = } 0,1,...,n-1$$

$$P_{r_{n-i}} = \frac{\lambda_r}{\mu_r} \frac{\lambda_f}{\lambda_r + (n-i)\lambda} P_{n-i}, \text{ i = } 0,1,...,n-1$$

$$P_{n-i} = \left(\frac{\lambda_f}{\mu}\right)^i \prod_{k=0}^{i-1} \left(1 - \frac{\lambda_f}{\lambda_r + (n-k)\lambda}\right) * P_n, \text{ i = } 1,2,...,n$$

$$P_n = \left[1 + \sum_{i=1}^{n} \left(\left(\frac{\lambda_f}{\mu}\right)^i \prod_{k=0}^{i-1} \left(1 - \frac{\lambda_r}{\lambda_r + (n-k)\lambda}\right)\right) + \left(\sum_{i=1}^{n-1} \frac{\lambda_f}{\lambda_r + (n-i)\lambda} \left(\frac{\lambda_f}{\mu}\right)^i \prod_{k=0}^{i-1} \left(1 - \frac{\lambda_r}{\lambda_r + (n-k)\lambda}\right) + \frac{\lambda_f}{\lambda_r + n\lambda}\right)\right]^{-1}$$

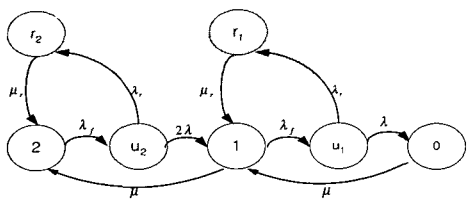


그림 2 이중계 HSS 시스템의 상태 모델

즉 시스템 운영 파라미터를 이용하여 P_n을 계산하면, 그 외의 모든 normal state(P_{n-i}), unstable state($P_{u_{n-i}}$) 및 rejuvenation state($P_{r_{n-i}}$)에서의 확률을 차례로 계산할 수 있다. 그림 2는 서버의 중복도를 2로 할 경우 HSS 시스템 운영 상태 모델의 예이며 시스템의 가동이 평형 상태일 때의 확률은 다음과 같다.

1) normal state :

$$P_2 + P_1 = \left(1 + \frac{\lambda_f}{u}\left(1 - \frac{\lambda_f}{\lambda_r + \lambda}\right)\right) P_2$$

2) unstable state :

$$P_{u_2} + P_{u_1} = \frac{\lambda_f}{\lambda_r + 2\lambda} P_2 + \frac{\lambda_f}{\lambda_r + \lambda} P_1$$

3) rejuvenation state :

$$P_{r_2} + P_{r_1} = \frac{\lambda_r}{\mu_r}\left(\frac{\lambda_f}{\lambda_r + 2\lambda}\right) P_2 + \frac{\lambda_r}{\mu_r}\left(\frac{\lambda_f}{\lambda_r + \lambda}\right) P_1$$

4) failure state :

$$P_0 = \frac{\lambda}{\mu} \frac{\lambda_f}{\lambda_r + \lambda} P_1$$

unstable state에서 가동되고 있는 2 대의 서버(u_2)는, 2λ의 변화율로 하드웨어적인 고장이 발생하며, 이 경우 λ는 서버의 MTTF로부터 계산할 수 있다. 두 대의 서버가 모두 고장인 상태인 failure state에서는 μ의 변화율로 고장이 수리되며, 이 값은 고장 수리 능력의 척도인 MTTR 에서 구할 수 있다. 서버의 장시간 가동으로 인한 소프트웨어 노화로 인해 서버의 성능이 저하되는 unstable state에 있을 경우, 고의로 시스템의 가동을 멈추는 rejuvenation state(r_2, r_1)로 가거나 혹은 고장이 발생하게 된다.

그림 3은 CSS 시스템의 운영상태 모델을 나타내며, 이 경우 특정 시점에서 가동되고 있는 서버는 오직 한 개이기 때문에 unstable state에서 고장 발생률(λ)이 일정하게 유지되도록 모델링 된다.

그림 3의 평형 상태에서의 balance equation은 다음과 같다.

$$\lambda_f P_n = \mu_r P_{r_n} + \mu P_{n-1}$$

$$(\lambda_f + \mu) P_{n-i} = \lambda P_{u_{n-i+1}} + \mu_r P_{r_{n-i}} + \mu P_{n-i-1}, \text{ i = } 1,2,...,n-1$$

$$(\lambda_r + \lambda) P_{u_{n-i}} = \lambda_f P_{n-i}, \text{ i = } 0,1,...,n-1$$

$$\mu_r P_{r_{n-i}} = \lambda_r P_{u_{n-i}}, \text{ i = } 0,1,...,n-1$$

$$\mu P_0 = \lambda P_{u_1}$$

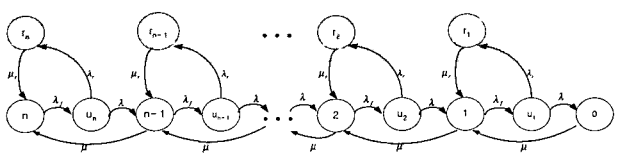


그림 3 소프트웨어 재활을 고려한 CSS 시스템 상태 모델

$$\sum_{i=0}^{n} P_i + \sum_{i=1}^{n} (P_{u_i} + P_{r_i}) = 1$$

또한 위의 연립 방정식을 풀면, 다음과 같은 평형상태에서의 확률을 얻을 수 있다.

$$P_{u_{n-i}} = \frac{\lambda_f}{\lambda_r + \lambda} P_{n-i},\ i = 0,1,\cdots,n-1$$

$$P_{r_{n-i}} = \frac{\lambda_r}{\mu_r} \frac{\lambda_f}{\lambda_r + \lambda} P_{n-i},\ i = 0,1,\cdots,n-1$$

$$P_{n-i} = \left(\frac{\lambda_f}{\mu} (1 - \frac{\lambda_f}{\lambda_r + \lambda}) \right)^i P_n,\ i = 1,2,\cdots,n$$

$$P_n = \left[1 + \sum_{i=1}^{n} \left(\frac{\lambda_f}{\mu} \left(1 - \frac{\lambda_r}{\lambda_r + \lambda} \right) \right)^i + \left(1 + \frac{\lambda_r}{\mu_r} \right) \left(\frac{\lambda_f}{\lambda_r + \lambda} \right) \sum_{i=0}^{n-1} \left(\frac{\lambda_f}{\mu} \left(1 - \frac{\lambda_r}{\lambda_r + \lambda} \right) \right)^i \right]^{-1}$$

4. 실험 및 성능 평가

① 가용도

HSS시스템에서 재활을 고려한 경우의 가용도(AvailHss)는 모든 서버가 고장 상태(P_0)이거나 한 대의 서버만이 가동중일 때 재활 작업을 수행(P_{r_1})하는 상태를 제외한 경우를 고려함으로써 계산될 수 있다.

$$\text{AvailHss} = 1 - (P_0 + P_{r_1})$$

$$\text{UnavailHss} = 1 - \text{AvailHss} = P_0 + P_{r_1}$$

CSS 시스템에서 재활을 고려한 경우의 가용도 계산은 다음과 같다.

$$\text{AvailCss} = 1 - (P_0 + \sum_{i=1}^{n} P_{r_i})$$

$$\text{UnavailCss} = P_0 + \sum_{i=1}^{n} P_{r_i}$$

② Downtime

HSS와 CSS 시스템의 재활 작업으로 인해 서비스를 받을 수 없는 downtime은 가동시간(T)에 대한 함수로 다음과 같다.

DownHss(T) = UnavailHss * T

DownCss(T) = UnavailCss * T

③ 손실 비용

HSS와 CSS 시스템의 가동 시간에 대한 손실 비용 함수는 다음과 같이 정의된다. 서버의 가동 정지로 인한 단위 시간당 손실 비용을 C_f, 재활 작업으로 인한 단위 시간당 손실 비용을 C_r이라 할 경우, 서버의 가동 정지로 인해 발생하는 비용은 다음과 같다. 일반적으로 예상 가능한 시스템 정지 비용은 불시 정지로 인한 손실 비용에 비해 훨씬 저렴하게 된다.($C_f >> C_r$)

$$\text{CostHss(T)} = (P_0 * C_f + P_{r_1} * C_r) * T$$

$$\text{CostCss(T)} = (P_0 * C_f + \sum_{i=1}^{n} P_{r_i} * C_r) * T$$

멀티미디어 이동 서비스의 특성상 가용도가 특히 중요시되는 컴퓨팅 시스템에 적합한 재활 정책을 개발하기 위하여, 가동 시간, 가동 방법, 고장률, 수리률 및 발생 비용 등의 변수를 복합적으로 고려하였다. downtime의 길이에 대한 고려보다는 이로 인해 발생하는 손실 비용이 더욱 중요한 요소가 되기 때문에 다양한 시스템 운영 파라미터를 동시에 고려하여 가용도 및 손실 비용을 계산하였다. 다양한 입력 파라미터에 대한 실험 결과의 평가 기준이 되는 시스템의 운영 파라미터는 그림 4와 같다[14]. HSS 혹은 CSS 방식으로 가동되는 서버 2대의 가동기간(T)은 1년으로 하며, 서버의 고장(λ)은 1년에 1회, 고장 수리(μ)에 1시간이 소요되고, 한 달에 한번씩 재활 작업(λ_r)을 수행하며, 이때 30분(μ_r)이 소요된다. 시스템의 불시정지로 인해 발생하는 비용(C_f)은 재활 작업으로 발생하는 비용(C_r)의 1000배가 된다.

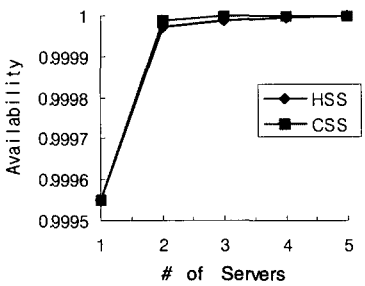
n= 2; number of servers
T = 12 * 30 * 24 * 3600; 1 year
λ = 1 / (12 * 30 * 24 * 3600); 1 time per year
μ = 1 / 3600; 1 time / hour
λ_r = 1 / (30 * 24 * 3600); 1 time per month
λ_f = 1 / (3 * 24 * 3600); every 3 days
μ_r = 2 / 3600; 2 times per hour
C_f = 1; downtime cost
C_r = 0.001; rejuvenation cost

그림 4 기준이 되는 시스템 운영 파라미터

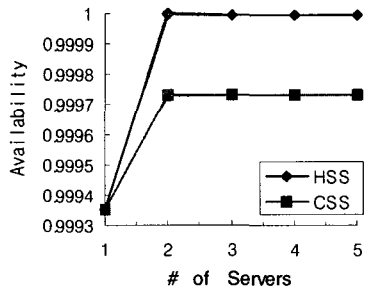
그림 5는 서버의 중복도에 따른 가용도, 손실 비용 및 downtime의 변화를 표시한다. 가용도의 경우 소프트웨어 재활을 하지 않은 경우((a) 참조)와 재활 작업을 수행한 경우((b) 참조)로 나누어지며, HSS 시스템의 경우 재활 작업을 수행하는 것이 더 우수했으나, CSS 시스템의 경우 재활을 하지 않는 것이 오히려 더 높게 나왔다. 이는 재활을 수행할 경우 CSS 시스템의 손실 비용 및 downtime이 더 높다는 것을 의미한다((c), (d) 참조). 단일계(n=1)로 시스템을 운영하는 경우에 비해 이중계(n≥2) 이상으로 시스템을 구성하면 가용도 및 손실 비용 측면에서 훨씬 유리하지만 n이 3 이상일 경우 그 상승폭은 미미했다. 따라서 대부분의 서버 가동의 경우 이중계로 구성하면 비용 효율적임을 알 수 있다.

HSS 시스템에서는 재활을 빈번히 할수록 가용도가 높아지지만, CSS 시스템의 경우에는 재활 작업은 바로 서비스의 중단을 의미하기 때문에 가용도가 재활률에 반비례하고 있는 추세를 나타낸다(그림 6 (a) 참조). 하지만 downtime으로 인해 발생하는 비용은 재활을 많이 하면 할수록 감소하는 추세를 보였으며(그림 6 (b) 참조), 두 시스템 모두 손실 비용이 증가하지 않는 이유는 불시에 발생하는 고장보다는 재활 작업으로 인한 시스템 정지로 발생하는 가용도 감소가 대부분을 차지하고 있기 때문이며, 불시의 고장으로 인해 발생하는 손실 비용이 재활률이 높아짐에 따라 감소함을 알 수 있다. 하지만 CSS 시스템의 주서버에서 여분서버로 switchover 할 경우에는 재활 작업으로 인해 서비스가 단절되므로, 빈번한 재활 작업은 HSS와는 달리 손실비용을 증가시킬 수도 있다.

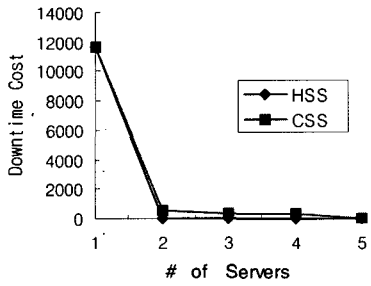
그림 7에서는 이중계 시스템의 경우 서버의 하드웨어적인 고장률 변화에 HSS와 CSS 시스템의 가용도는 거의 무관함을 나타내고 있다. (b)의 경우 고장률이 0.33(10일에 한번씩 고장)일 경우를 기준으로 이보다 작을 경우에는 HSS 시스템이 이보다 더 클 경우에는 CSS 시스템의 손실 비용이 적게 들고 있음을 나타낸다. 이는 시스템의 운영 방식 결정에 하드웨어의 고장률이 중요한 요소로 작용하고 있음을 나타낸다. 그림 8은 normal state에서 unstable state로의 변화에 대한 가용도 및 손실 비용 관계를 나타내고 있다. HSS 시스템의 경우 가용도는 거의 변화가 없었으며((a) 참조), 변화율이 0.5(시스템이 가동 후 12시간이 지나면 불안정 상태로 됨) 이상일 경우, CSS 시스템 보다 비용이 더 높아졌다. 고장률과 마찬가지로 unstable state로의 변화율이 재활 정책 수립의 중요한 요소임을 파악할 수 있다. 그림 9는 서버의 수리률 변화에 대한 가용도 및 손실 비용의 추세를 나타내고 있으며, 시스템의 수리하는 능력, 즉 수리률이 높아질수록 시스템 가용도는 높아지고, 손실 비용은 감소하는 결과를 얻을 수 있었다. 수리률 1(시스템 고장 발생 후 24시간이 지나면 가동 상태가 됨)을 기준으로 HSS 시스템의 손실 비용이 CSS 시스템의 손실 비용보다 적어지는 경향을 나타냈다. CSS 시스템의 경우 수리률이 작으면 작을수록 손실 비용 측면에서 HSS에 비해 유리하다. 그림 10은 재활 작업에 소요되는 시간의 변화에 대한 가용도 및 손실 비용의 관계를 나타내며, CSS 시스템의 경우 재활 소요 시간에 민감한 반응을 하고 있음을 알 수 있다. 이는 HSS 시스템에서는 재활 작업 중에 여분의 프로세스가 계속해서 서비스를 제공하고 있는 반면에 CSS 시스템의 경우



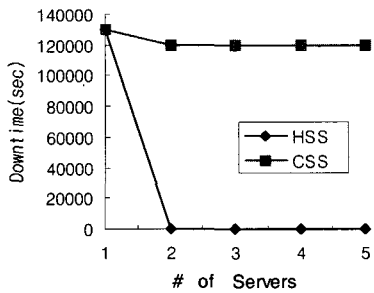
(a) 소프트웨어 재활이 없는 경우



(b) 소프트웨어 재활을 수행한 경우

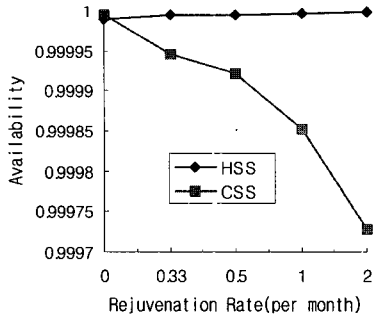


(c) 손실 비용

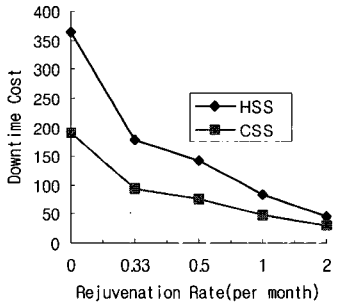


(d) Downtime

그림 5 중복도(n)에 따른 가용도, 손실 비용 및 Downtime의 변화

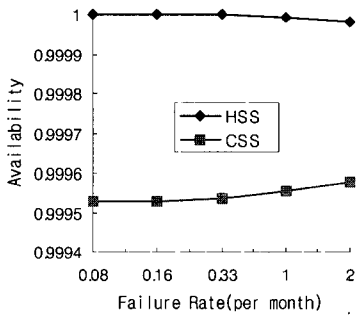


(a) 가용도

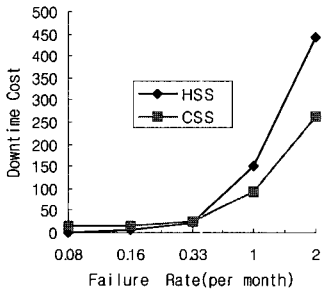


(b) 손실 비용

그림 6 재활률(λ_r)에 대한 가용도 및 손실 비용의 변화

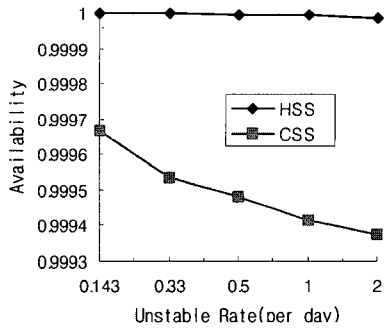


(a) 가용도

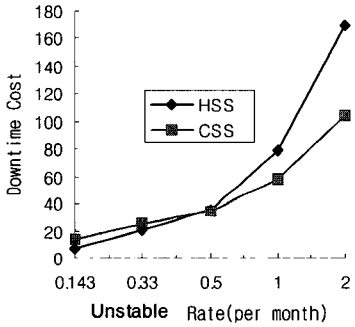


(b) 손실 비용

그림 7 고장률(λ)에 대한 가용도 및 손실 비용의 변화



(a) 가용도

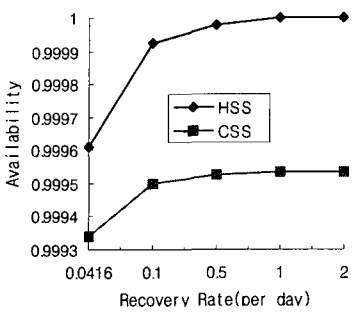


(b) 손실 비용

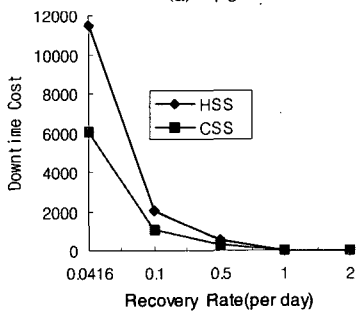
그림 8 불안정률(λ_f)에 대한 가용도 및 손실 비용의 변화

재활 작업 중에는 서비스가 중단됨으로 인하여 발생한 것으로 판단된다.

CSS 시스템의 switchover state를 고려하지 않은 본 연구에서는, HSS와 CSS 시스템 모두 재활을 자주 할수록 가용도는 증가하고 손실 비용이 감소되는 것을 확인하였으며 이를 통해 재활 기법의 적용 가능성이 높다고 판단할 수 있었다. 단일계와 다중계 시스템을 비교할 경우, 당연히 다중계로 구성된 시스템의 가용도 증가로 인하여 시스템 결함으로 발생하는 비용은 줄어드나, n ≥3 시스템에서는 가용도 증가율이 그리 크지 않았다. 다중계 시스템에서 비교를 할 경우 이중계일 때는 HSS > HSSR > CSSR > CSS 순으로 손실 비용이 파악되었으므로(여기서 R이 붙은 경우, 소프트웨어 재활의 수행을 의미함), HSS 시스템의 경우 재활 수행이 유리하고 CSS시스템의 경우, 재활 작업 수행시 불리하게 된다. n≥3일 때는 CSSR > HSS > HSSR > CSS로 손실 비용이 발생함으로, HSS시스템에서는 항상 재활

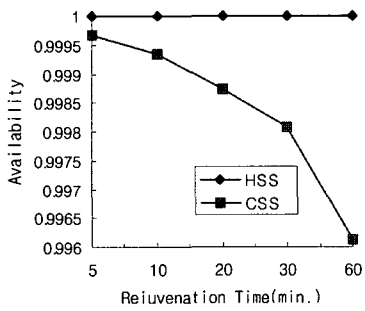


(a) 가용도

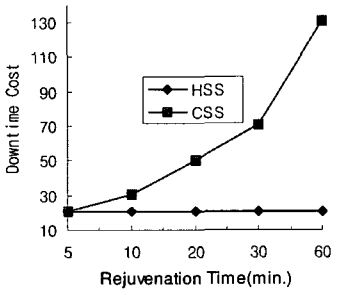


(b) 손실 비용

그림 9 서버의 수리률(μ)에 대한 가용도 및 손실 비용의 변화



(a) 가용도



(b) 손실 비용

그림 10 재활 작업 시간($1/\mu_r$)에 대한 가용도 및 손실 비용의 변화

을 고려하는 것이 유리하며, CSS시스템에서는 재활 작업을 안 하는 것이 유리하다. 따라서 원하는 수준의 가용도와 지불할 수 있는 비용 여력에 맞는 시스템을 구성할 수 있으며, 이때 고장률과 불안정률에 따라 시스템의 운영을 HSS로 할 것인가 CSS로 할 것인가에대한 의사결정이 가능하게 되며, 최적 재활 주기의 결정은 가용도와 손실 비용을 동시에 고려하는 방식으로 결정되어야 할 것이다. 시스템을 가동시켜 얻은 실측치가 아닌 수학적 모델에 근거한 데이타를 해석한 본 논문의 결과는 본 모델에서 제시한 방법대로 시스템을 운영한 후 실제 운영 데이타를 얻으면, 그 정확성이 판단될 수 있으리라 본다.

5. 결 론

본 연구에서는 서버에서 수행되는 소프트웨어의 재활 주기, 재활 소요시간, 서버의 고장률, 수리률, 동시에 가동되는 서버의 수, 서버의 가동 기간 및 가동 방식 등의 시스템 운영 파라미터에 기초하여, 소프트웨어 재활 정책에 대한 평가를 위한 평형 상태에서의 확률, downtime, 가용도, 손실 비용 등을 계산하였다. 즉 고장률이 0.33(10일에 한번씩 고장 발생)인 경우와 불안정률이 0.5(초기 가동후 15일 후면 시스템의 고장 가능성이 높아짐) 이상인 경우에 손실 비용 측면에서 시스템 운영 방식이 바뀌는 것을 확인하였다. 수학적 분석을 통해 구한 재활 모델의 closed-form 해는 다양한 시스템 운영 상태에 대한 실험을 통해 검증하였으며, 소프트웨어 재활을 통한 예방적 결함허용 기법의 적용 가능성이 높다는 것을 확인하였다. 또한 서버의 고장률 및 불안정률이 소프트웨어 재활 정책 결정에 중요한 요소임을 파악하였다. 멀티미디어 이동 컴퓨팅에서 사용되는 소프트웨어는 통신 단절, 데이타 유실에 의한 노화가 빨리 진행되어 결함 발생 가능성이 다른 시스템에 비해 현저히 높다. 따라서 소프트웨어 재활 방법은 컴퓨터 자원의 추가 비용이 없고, 최근 시스템 비용 중 가장 커다란 비중을 차지하는 소프트웨어 유지보수 비용을 상당히 줄일 수 있는 최선의 해결책으로 판단된다.

추후에는 시스템에 고장이 발생할 경우, 이를 검출해 내는 능력인 coverage factor 및 switchover time을 고려할 예정이고 그밖에 각 서버 내에서 요청 작업에 대한 처리 능력을 나타내는 throughput, 작업 응답 시간과 비용 함수를 재활 정책(policy)과 결합시키는 방법에 대한 연구가 요청된다. 소프트웨어 최적 재활 주기를 산정할 때에, 시스템의 부하를 고려해서 정책을 수립할 필요성이 있고 checkpointing방법을 소프트웨어 재활 기

법과 결합시킬 경우에 평균 완료시간(expected completion time)을 더욱 더 감소시킬 수 있을 것으로 기대된다.

참 고 문 헌

[1] 김대영, "차세대 인터넷 기술 동향", 정보과학회지 제17권 제3호, pp. 4-13, 1999. 3.

[2] 권호열, 장은정, "고속 LAN과 가입자망 기술의 연구 동향", 정보과학회지 제17권 제4호, pp. 4-13, 1999. 4.

[3] N. Talagala and D. Patterson, "An analysis of error behavior in a large storage system," IEEE Workshop on Fault-Tolerant Parallel and Distributed Systems, pp. 28-51, San Juan, Puerto Rico, Apr. 1999.

[4] R. Jain, *The Art of Computer Systems Performance Analysis.* p. 685, John Wiley & Sons Inc., 1991.

[5] 김춘길, "전사상거래의 개념과 발전방향", 정보과학회지 제16권 제5호, pp. 5-10, 1998. 5.

[6] 김진상, 박재희, 방갑산, "ERP 기술개발 동향 및 추세", 정보과학회지 제16권 제11호, pp. 6-12, 1998. 11.

[7] I. Lee and R. Iyer, "Software dependability in the Tandem GUARDIAN system," IEEE Transactions on Software Engineering, Vol. 21, No. 5, pp. 455-467, May 1995.

[8] B. Johnson, *Design and Fault-Tolerant Analysis of Digital Systems.* p. 584, Addison-Wesley Publishing Company, 1989.

[9] A. Pfening, S. Garg, M. Telek, A. Puliafito and K. Trivedi, "Optimal rejuvenation for tolerating soft failures," Performance Evaluation, Vol. 27 & 28, North-Holland, pp. 491-506, Oct. 1996.

[10] S. Garg, Y. Huang, C. Kintala and K. Trivedi, "Time and load based software rejuvenation: policy, evaluation and optimality," Proc. of the first conference on Fault tolerant systems, Madras, India, Dec. 1995.

[11] S. Garg, A. Puliafito, M. Telek and K. Trivedi, "Analysis of software rejuvenation using Markov regenerative stochastic Petri net," Proc. of the Sixth International Symposium on Software Reliability Engineering, pp. 180-187, Toulouse, France, Oct. 1995.

[12] S. Garg, A. Puliafito, M. Telek and K. Trivedi, "On the analysis of software rejuvenation policies," Proc. 12th Annual Conference on Computer Assurance (COMPASS), June 1997.

[13] S. Garg, A. Puliafito, M. Telek and K. Trivedi, "Analysis of preventive maintenance in transactions based software systems," IEEE Transactions on Computers, Vol. 47, No. 1, pp. 96-107, Jan. 1998.

[14] Y. Huang, C. Kintala, N. Kolettis and N. Fulton, "Software rejuvenation: analysis, module and applications," Proceedings of the 25th International Symposium on Fault Tolerant Computing (FTCS-25), Pasadena, CA, pp. 381-390, June 1995.

[15] Y. Wang, Y. Huang, W. Fuchs, C. Kintala and G. Suri, "Progressive retry for software failure recovery in message-passing applications," IEEE Trans. on Computers, Vol. 46, No. 10, pp. 1137-1141, Oct. 1997.

[16] Y. Huang, C. Kintala, L. Bernstein and Y. Wang, "Components for software fault tolerance and rejuvenation," AT&T Technical Journal, pp. 29-37, Mar. 1996.

[17] Y. Huang, C. Kintala and Y. Wang, "Software tools and libraries for fault tolerance," Bulletin of the Technical Committee on Operating Systems and Application Environment (TCOS), Vol. 7, No. 4, pp. 5-9, Winter 1995.

[18] S. Garg, Y. Huang, C. Kintala and K. Trivedi, "Minimizing completion time of a program by checkpointing and rejuvenation," Proc. 1996 ACM SIGMETRICS Conference, pp. 252-261, Philadelphia, PA, May 1996.

[19] H. Levendel, "Software dependability in wireless systems," IEEE Workshop on Fault-Tolerant Parallel and Distributed Systems, pp. 3-12, San Juan, Puerto Rico, Apr. 1999.

[20] K. Trivedi, *Probability and Statistics with Reliability, Queueing, and Computer Science Applications.* p. 624, Prentice-Hall, 1982.

[20] L. Kleinrock, *Queueing Systems Volume1: Theory.* p. 417, John Wiley & Sons Inc., 1975.

[22] G. Thomas and R. Finney, *Calculus and Analytic Geometry.* p. 891, Addison-Wesley Publishing Company, 1979.

박 기 진

1989년 한양대학교 산업공학과(공학사). 1991년 포항공과대학교 산업공학과(공학석사). 1991년 ~ 1996년 삼성종합기술원 기반기술연구소 선임연구원. 1996년 ~ 1997년 삼성전자(주) 소프트웨어센타 선임연구원. 1997년 ~ 현재 아주대학교 컴퓨터공학과 박사과정. 관심분야는 멀티미디어 시스템, 결함허용 시스템, 성능 평가, 시뮬레이션

김 성 수

1992년 서강대학교 전자공학과(공학사). 1984년 서강대학교 전자공학과(공학석사). 1995년 Texas A&M University, 전산학과(공학박사). 1983년 ~ 1996년 삼성전자(주) 종합연구소 컴퓨터연구실(주임연구원). 1986년 ~ 1996년 삼성종합기술원 수석연구원. 1991년 ~ 1992년 Texas Transportation Institute 연구원. 1993년 ~ 1995년 Texas A&M University, 전산학과, T.A. 1997년 ~ 1998년 한국정보과학회, 한국정보처리학회 논문지 편집위원. 1996년 ~ 현재 아주대학교 정보통신전문대학원 부교수. 관심분야는 멀티미디어, 성능 평가, 결함 허용, 이동 컴퓨팅, 시뮬레이션.

김 재 훈

1984년 서울대학교 제어계측공학과(학사). 1993년 Indiana University 전산학과(석사). 1997년 Texas A&M University 전산학과(박사). 1995년 ~ 1997년 Texas A&M University 전산학과 Graduate Research Assistant. 1984년 ~ 1991년 대우통신(주) 컴퓨터연구원. 1997년 ~ 1998년 삼성전자 컴퓨터시스템팀. 1998년 ~ 현재 아주대학교 정보통신전문대학원 조교수. 관심분야는 분산시스템, 실시간 시스템, 운영체제, 결함허용.