소프트웨어 재활기법에 기반한 주-여분 서버 시스템의 작업전이 시간 분석

이 재 성[†] · 박 기 진[††] · 김 성 수[†††]

요 약

인터넷의 급속한 확산으로 인하여, 컴퓨터 시스템의 규모 및 복잡도가 점차 증가하고 있으며, 컴퓨터 시스템에 대한 높은 수준의 가용도 요구 조건을 충족시키기 위해, 하드웨어적·소프트웨어적 결함 허용 기법에 대한 연구가 활발하다. 소프트웨어 재활 기법은 서버에 탑재된 소프트웨어의 장시간 가동에 따른 소프트웨어 노화 현상을 다루고 있으며, 서버에서 실행중인 소프트웨어의 수행을 고의적으로 멈춘 후에 결함 발생 가능성이 낮은 초기 상태에서 이를 다시 가동시키는 소프트웨어적 결함 예방 방법의 일종이라 볼 수 있다. 본 연구에서는 주-여분 서버 시스템에서의 작업전이 시간이 소프트웨어 재활에 미치는 영향을 연구하였으며, 가용도 분석을 통해서 작업전이 시간이 재활 정책을 결정함에 있어서 중요한 요소임을 발견하였다.

Switchover Time Analysis of Primary-Backup Server Systems Based on Software Rejuvenation

Chae-Sung Lee[†] · Kie-Jin Park[††] · Sung-Soo Kim[†††]

ABSTRACT

As the rapid growth of Internet, computer systems are growing in its size and complexity. To meet high availability requirements for the systems, one usually uses both hardware and software fault tolerance techniques. To prevent failures of computer systems from software-aging phenomenon that come from long mission time, we adopt software rejuvenation method that stops and restarts the software in the servers intentionally. The method makes the systems clean and healthy state in which the probability of fault occurrence is very low. In this paper, we study how switchover time affects software rejuvenation of primary-backup server systems. Through experiments, we find that switchover time is an essential factor for deciding the rejuvenation policy.

키워드 : 소프트웨어 재활(Software Rejuvenation), 가용도(Availability), 결함허용(Fault-tolerance)

1. 서 론

인터넷의 성장과 인터넷 이용자의 증가로 인해서 컴퓨터 시스템은 점차 대규모화·복잡화되고 있다. 인터넷 환경에서는 전자 상거래와 같은 인터넷을 이용한 서비스의 증가 및 인터넷 사용자 수의 증가가 예상되며, 이에 따른 서비스 요구를 만족시켜주기 위해 가용도 요구 수준이 매우 엄격한 고속·대용량 시스템의 보급 확산이 빠르게 이루어지고 있다. 따라서 현대 사회에서는 컴퓨터 시스템에 대한 의존도가 계속해서 높아지고, 컴퓨터 시스템의 다운으로 인한 피해도 증가하는 추세이기 때문에 컴퓨터 시스템에 대한 가용도가 더욱더 중요시 될 것으로 예측된다[1-3].

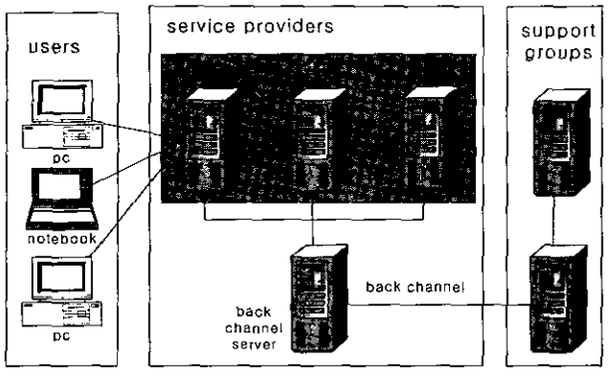
하드웨어적으로 가용도를 높이기 위한 방법으로는 주서버(primary server)를 대체하는 여분서버(spare server)를 두는 방법이 사용되며, 이것은 주서버에서 여분서버로의 작업전이 방법에 따라서 CS(cold standby)방법과 HS(hot standby)방법으로 구분하며 이러한 방법을 이용해서 구성한 시스템을 각각 CS 시스템과 HS 시스템이라고 한다. HS 시스템은 주서버를 대체하는데 필요한 시간은 매우 짧지만 모든 서버를 동시에 돌려야 하므로 많은 비용이 들고 서버의 결함 발생률 또한 높은 반면에, CS 시스템은 비록 주서버를 대체할 때까지 걸리는 시간은 길지만 비용이 적게 들고 서버의 결함 발생률도 낮다.

높은 수준의 가용도 요구는 인터넷 컴퓨팅 환경에서는 반드시 선행되어야할 요건이라 할 수 있다. 인터넷 사용자는 언제든지 인터넷을 통해서 서비스 요청을 할 수 있어야 하기

※ 정보통신부에서 지원하는 대학기초연구지원 사업으로 수행.
† 준 회 원 : 아주대학교 정보통신전문대학원
†† 준 회 원 : 한국전자통신연구원(ETRI) 네트워크장비시험센터 선임연구원
††† 정 회 원 : 아주대학교 정보통신전문대학원 교수
논문접수 : 2000년 11월 13일, 심사완료 : 2001년 5월 17일

때문에, 인터넷 서비스 시스템은 하루 24시간 365일 계속해서 작동되어야만 한다[4, 5]. 이처럼 높은 수준의 가용도를 제공하기 위해 CS 시스템과 HS 시스템 중 어느 하나를 선택할 수 있으며, 이 때 중요한 요소는 서버의 작업전이 시간이다. 일반적으로, 높은 수준의 가용도를 요구하는 시스템에서는 HS 시스템을 사용하는 것이 유리하고 상대적으로 낮은 수준의 가용도를 요구하는 시스템에서는 CS 시스템을 사용하는 것이 바람직하다. 또한 높은 수준의 가용도를 요구하지만 비용적인 측면도 고려해야 한다면 CS 시스템을 사용하는 것이 HS 시스템보다 좋다. 예를 들면, 인터넷을 이용한 서비스 중에서 전자 상거래 시스템에 CS 시스템을 이용할 수 있다. 왜냐하면 CS 시스템은 작업전이 시간은 상대적으로 길지만, 작업전이 시간이 서비스를 요청하는 빈도수가 가장 적은 순간에 이루어지도록 해준다면 HS 시스템과 비교해서 작업전이 시간으로 발생하는 손실에서 큰 차이를 보이지 않을 수 있고, 비용적인 측면에서도 유리하기 때문이다.

최근에는 기술의 발전으로 하드웨어에 의한 결함 발생률은 점차로 줄어들고 있으며, 오히려 소프트웨어에 의한 결함이 전체 시스템 결함의 62%를 차지하는 것으로 보고되었다[6-8]. 소프트웨어의 결함을 줄이기 위한 방법으로 가장 이상적인 것은 결함이 없는(fault-free) 소프트웨어를 만드는 것이지만, 결함이 없는 소프트웨어를 개발하는 것은 불가능하기 때문에 기존의 방법들은 결함이 발생한 후에 이를 해결하고 있다. 반면에, 소프트웨어 재활은 소프트웨어적인 결함을 사전에 막아서 결함의 원인을 없애주는 방법으로 시스템 관리자의 입장에서 보면 시스템을 효율적으로 운영할 수 있게 해주는 좋은 방법 중 하나라고 판단된다.



(그림 1) 다중 서버로 구성된 CS 시스템

(그림 1)은 다중 서버로 구성된 CS 시스템의 구성 예이다. (그림 1)을 보면 사용자는 인터넷을 통해 서비스 제공자의 주서버에 접근해서 서비스를 요청하고 그에 대한 응답을 받는다. 이 때 모든 사용자의 요구를 수행하는 주서버에서 실행 중에 결함이 발생하면 여분서버가 주서버의 작업을 대신해서 실행하며, 거래에 필요한 여러 가지 다른 정보들은 다른 서비스 제공자의 시스템과 연결되어 있는 back channel 서버를 통해서 얻는 방식으로 작동한다. 본 연구에서는 (그림 1)의 회색영역에 있는 서버를 대상으로 가동중인 주서버와 주서버에 결함이 발생했을 때만 가동되는 다수의 여분서버들로 구성된 CS 시스템에서, 소프트웨어 재활 기법을 통한 가용도 개선과 손실비용을 줄이기 위한 재활 정책을 연구하였다.

본 논문의 2장에서는 소프트웨어 재활 기법과 기존의 연구들에 대해 논하였으며, 3장에서는 소프트웨어 재활 기법을 적용한 CS 시스템에서, 서버의 작업전이 시간을 고려한 모델을 제안하였고, 4장에서는 3장에서 제안한 모델을 수학적 해석 방법의 정확성을 검증하기 위해서 다양한 시스템 운영 조건에 대한 실험을 수행하였다. 마지막으로 5장에서는 CS 시스템에서의 소프트웨어 재활 기법과 향후 연구에 대해 논하였다.

2. 관련 연구

소프트웨어를 오랜 기간동안 계속해서 실행시키면 메모리 부족, 데이터의 불일치, 저장 공간의 단편화 현상과 같은 원인으로 인한 노화현상이 발생하며, 이로 인해 서버에서 실행중인 소프트웨어가 불안정한 상태에서 동작하기 때문에, 결국 컴퓨터 시스템에 결함을 일으킨다. 컴퓨터 시스템의 결함을 발생시키는 원인이 되는 노화 현상을 제거함으로써 결함 발생을 사전에 막아보려는 방법이 소프트웨어 재활이며, 이는 현재 서버에서 실행중인 소프트웨어가 불안정한 상태로 실행하고 있을 때 이를 초기상태로 환원시킨 후 다시 실행시키는 것으로, 노화현상의 발생을 막고 이로 인한 컴퓨터 시스템의 결함 발생 횟수를 감소시켜 준다[9-15].

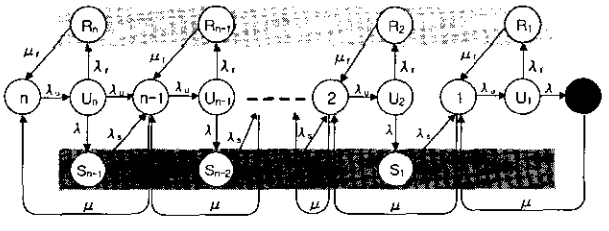
서버가 1대로 구성된 시스템[16-21]과 서버가 2대 이상으로 구성된 시스템[22, 23]에 대하여 소프트웨어 재활 기법을 적용한 기존 연구 사례가 있었으며, 서버가 2대 이상으로 구성된 기존 연구에서는 CS 시스템에서의 재활에 대한 가용도와 손실비용 및 재활정책에 관한 논의가 있었고, [23]에서는 서버들간의 작업전이 시간을 고려하였다. 하지만, [23]에서는 단지 작업전이 시간을 포함한 모델을 가지고 재활에 대한 언급만 하였을 뿐 실제로 작업전이 시간에 대한 영향은 논하지 않았다. CS 시스템에서 작업전이 시간은 여분서버의 파워를 켜고 여분서버가 주서버를 대신해 서비스를 다시 시작 할 때까지 걸리는 시간으로 재활작업 시간과 거의 같은 시간이 걸린다고 볼 수 있기 때문에, HS 시스템과는 달리 서버의 작업전이에 상대적으로 긴 시간이 필요하며, 소프트웨어 재활을 했을 경우와 하지 않았을 경우를 비교할 때, CS 시스템의 작업전이 시간은 실제로 상당히 긴 시간에 속한다. 이는 소프트웨어 재활을 하는데 걸리는 시간은 최악의 경우 서버를 끄고 다시 켠다고 하더라도 작업전이 시간이하가 걸리기 때문이다. 본 연구에서는 CS 시

스템의 모델링을 통해서, 작업전이 시간이 재활주기 또는 손실비용 척도 및 전체 시스템의 재활정책에 미치는 영향에 대한 분석을 하였다.

3. CS 시스템 모델

CS 시스템의 작업전이 시간을 고려한 상태 모델은 (그림 2)와 같으며, 다음의 가정들을 사용하였다.

- 재활작업 시간($1/\mu_r$)은 동일하다.
- 고장 난 서버를 수리하는 수리시간($1/\mu$)은 동일하다.
- n대의 서버로 구성된 시스템에서 각 서버의 고장률(λ)은 동일하다.
- 여분서버가 고장난 주서버를 대체하는 데 걸리는 작업전이 시간($1/\lambda_s$)은 동일하다.
- 재활을 할 때, 현재 가동 중인 주서버만이 재활 대상에 포함된다.
- 서버의 가동을 주기적으로 멈추는 재활률(λ_r)은 모든 가동 상태에서 동일하다.
- 모든 상태에 머무는 시간은 지수분포를 따른다.



(그림 2) CS 시스템에서 작업전이 시간을 고려한 상태모델

(그림 2)의 CS 시스템은 n대의 서버로 구성되어 있으며, 이 중 한 대를 제외한 n-1대의 서버들이 여분서버이다. {n, n-1, …, 2, 1}은 결함발생이 적은 정상 상태이며, {0}은 모든 서버의 결함으로 시스템이 다운된 고장 상태를 가리킨다. $\{U_n, U_{n-1}, \cdots, U_2, U_1\}$은 서버의 장시간 가동으로 인한 서버의 불안정 상태를 나타내며, $\{R_n, R_{n-1}, \cdots, R_2, R_1\}$은 서버의 가동을 강제적으로 정지시킨 후, 건강한 상태로 되돌리는 재활 상태를 의미한다. $\{S_{n-1}, S_{n-2}, \cdots, S_2, S_1\}$은 주서버의 고장으로 인한 서버들간의 작업전이 과정이 일어나는 작업전이 상태를 표시한다. 여기서 밑의 첨자들과 정상 상태에서의 숫자들은 정상적으로 작동이 가능한 서버 수를 가리킨다. 처음에 시스템은 정상 상태(i)에 머물다가 서버의 계속적인 가동으로 인해서 불안정 상태(U_i)로 바뀌며, 이 상태에서는 서버에 결함이 발생하기 전에 재활작업을 해주는 재활 상태(R_i) 또는, 결함 발생 후에 결함 허용을 수행하는 작업전이 상태(S_i)로 이동할 수 있다. 재활 상태에서는 재활작업 시간($1/\mu_r$) 후에 결함 발생이 적은 정상 상태(i)로 환원되며, 작업전이 상태에서는 일정 작업전이 시간($1/\lambda_s$) 후에 서버의 수가 한 대 줄어든 정상 상태(i-1)로 전이한다. 또한 고장난 서버는 수리 시간($1/\mu$) 후에 다시 정상적으로 가동될 수 있으며, 이때 시스템의 상태는 정상 상태를 유지하지만 정상적으로 작동 가능한 서버 수의 증가로 인해서 (i-1)에서 (i)로 바뀌게 된다. <표 1>에 각각의 상태들에 대한 정의를 요약하였다.

〈표 1〉 각각의 상태 정의

상태	상태 정의
i	가동할 수 있는 서버의 개수(1, 2, …, n-1, n)
U_i	서버가 계속해서 가동됨으로 인해 성능이 저하된 불안정 상태
R_i	서버가 재활작업을 하고 있는 재활 상태
S_i	고장난 서버를 여분서버로 대체하는 작업전이 상태
P_i	서버가 i 상태에 있을 확률
P_{u_i}	서버가 U_i 상태에 있을 확률
P_{r_i}	서버가 R_i 상태에 있을 확률
P_{s_i}	서버가 S_i 상태에 있을 확률

(그림 2)는 각 상태에서 머무는 시간이 지수분포를 따르는 Markov 프로세스로 해석되며, 평형 상태에서의 평형상태 방정식은 다음과 같다.

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

$$(\lambda_u + \mu) \cdot P_{n-i} = \mu_r \cdot P_{r_{n-i}} + \lambda_s \cdot P_{s_{n-i}} + \mu \cdot P_{n-i-1}, \quad i = 1, 2, \cdots, n-1$$

$$(\lambda + \lambda_r) \cdot P_{U_{n-i}} = \lambda_u \cdot P_{n-i}, \quad i = 0, 1, \cdots, n-1$$

$$\mu_r \cdot P_{r_{n-i}} = \lambda_r \cdot P_{u_{n-i}}, \quad i = 0, 1, \cdots, n-1$$

$$\lambda_s \cdot P_{s_{n-i}} = \lambda \cdot P_{u_{n-i+1}}, \quad i = 1, 2, \cdots, n-1$$

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

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

위의 방정식을 풀면 다음과 같은 평형 상태에서의 확률을 얻을 수 있다.

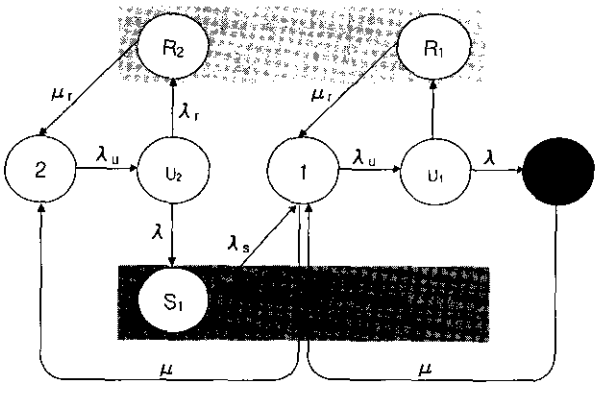
$$P_{u_i} = \frac{\lambda_u}{\lambda + \lambda_r} \cdot P_i, \quad i = 1, 2, \cdots, n$$

$$P_{r_i} = \frac{\lambda_r}{\mu_r} \cdot \frac{\lambda_u}{\lambda + \lambda_r} \cdot P_i, \quad i = 1, 2, \cdots, n$$

$$P_{s_i} = \frac{\mu}{\lambda_s} \cdot P_i, \quad i = 1, 2, \cdots, n-1$$

$$P_i = \left(\frac{\lambda \cdot \lambda_u}{\mu \cdot (\lambda + \lambda_r)} \right)^{n-i} \cdot P_n, \quad i = 0, 1, \cdots, n$$

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



(그림 3) 서버가 2대인 CS 시스템의 상태모델

(그림 3)은 2대의 서버로 구성된 CS 시스템의 상태 모델을 보여주고 있으며, 이 모델에 위에서 구한 해를 적용하면 다음과 같은 각각의 상태에서의 확률 값을 구할 수 있다.

$$P_{u_1}=\frac{\lambda_u}{\lambda+\lambda_r}\cdot P_1 \qquad P_{u_2}=\frac{\lambda_u}{\lambda+\lambda_r}\cdot P_2$$

$$P_{r_1}=\frac{\lambda_r}{\mu_r}\cdot\frac{\lambda_u}{\lambda+\lambda_r}\cdot P_1 \qquad P_{r_2}=\frac{\lambda_r}{\mu_r}\cdot\frac{\lambda_u}{\lambda+\lambda_r}\cdot P_2$$

$$P_{s_1}=\frac{\mu}{\lambda_s}\cdot P_1$$

$$P_0=\left(\frac{\lambda\cdot\lambda_u}{\mu\cdot(\lambda+\lambda_r)}\right)^2\cdot P_2 \qquad P_1=\left(\frac{\lambda\cdot\lambda_u}{\mu\cdot(\lambda+\lambda_r)}\right)^1\cdot P_2$$

$$P_2=\left[\sum_{i=1}^{2}\left(\frac{\lambda\cdot\lambda_u}{\mu\cdot(\lambda+\lambda_r)}\right)^{2-i}\cdot\left\{1+\frac{\lambda_u}{\lambda+\lambda_r}\left(1+\frac{\lambda_r}{\mu_r}\right)+\frac{\mu}{\lambda_s}\right\}+\left(\frac{\lambda\cdot\lambda_u}{\mu\cdot(\lambda+\lambda_r)}\right)^2-\frac{\mu}{\lambda_s}\right]^{-1}$$

$$=\left[\left\{1+\left(\frac{\lambda\cdot\lambda_u}{\mu\cdot(\lambda+\lambda_r)}\right)^1\right\}\cdot\left\{1+\frac{\lambda_u}{\lambda+\lambda_r}\cdot\left(1+\frac{\lambda_r}{\mu_r}\right)+\frac{\mu}{\lambda_s}\right\}+\left(\frac{\lambda\cdot\lambda_u}{\mu\cdot(\lambda+\lambda_r)}\right)^2-\frac{\mu}{\lambda_s}\right]^{-1}$$

위의 결과 값은 서버가 2대일 경우로 P_1, P_2는 정상 상태, P_{u_1}, P_{u_2}는 불안정 상태, P_{r_1}, P_{r_2}는 재활 상태, P_{s_1}은 작업전이 상태, P_0은 고장 상태를 의미한다. 각각의 상태 값들을 더해줌으로써 시스템의 5가지 상태에 대한 전체 확률 값을 얻을 수 있으며, 시스템 관리자는 이 값을 구함으로써 전반적인 시스템의 동작 상태를 예측할 수 있다. 이는 확률 값이 시스템의 전체 가동시간 동안 머무는 상대적인 확률을 내포하기 때문에 특정 상태에서의 확률 값이 다른 상태의 확률 값보다 크다면 시스템이 이 상태에서 더 많이 머문다는 것을 의미한다.

4. CS 시스템 모델 분석

모델링 분석을 통한 CS 시스템의 가용도와 손실비용을 다음과 같이 정의하였다.

- 소프트웨어 재활을 하지 않았을 경우의 가용도와 손실비용

$$availability = 1-\left(P_0+\sum_{i=1}^{n-1}P_{s_i}\right)$$

$$downtime\ cost=\left(P_0\times C_f+\sum_{i=1}^{n-1}P_{s_i}\times C_s\right)\times L$$

- 소프트웨어 재활을 했을 경우의 가용도와 손실비용

$$availability=1-\left(P_0+\sum_{i=1}^{n-1}P_{s_i}+\sum_{i=1}^{n}P_{r_i}\right)$$

$$downtime\ cost=\left(P_0\times C_f+\sum_{i=1}^{n-1}P_{s_i}\times C_s+\sum_{i=1}^{n}P_{r_i}\times C_r\right)\times L$$

가용도(availability)는 서버의 가동 유·무를 말하며, 모든 서버가 고장난 상태(P_0)와 주서버에 결함이 발생했을 때 여분서버로의 작업전이를 수행하는 작업전이 상태(P_{s_i}), 서버의 재활작업을 수행하는 재활 상태(P_{r_i})를 제외시킴으로써 구할 수 있다.

CS 시스템의 손실비용(downtime cost)은 서버의 결함에 기인한 시스템의 서비스 중지로 인해 발생하는 손실로, 서비스를 제공받을 수 없는 상태인 P_{r_i}, P_{s_i}, P_0에 각각의 단위시간당 손실비용인 C_r, C_s, C_f를 곱해서 단위시간당 총 손실비용을 구한 후, 이를 전체 시스템의 가동시간(L)과 곱함으로써 얻을 수 있다. 특히, C_r, C_s, C_f는 서버의 결함 발생으로 인한 단위시간당 손실비용으로 C_r값은 계획된 서버의 중지로 인해 발생하는 손실비용이므로 가장 낮은 값을 가지며, C_s는 갑작스런 서버의 결함으로 인해 작업전이 시간동안 발생하는 손실비용으로 C_r값보다는 높다. 이는 시스템이 S_1 상태로 전이가 일어나는 시점(주서버의 고장으로 여분서버로의 작업전이가 일어나는 시점)을 예측할 수 없기 때문이다. C_f는 갑작스런 서버의 결함으로 전체 시스템이 다운되었을 때 발생하는 손실로 가장 높다.

<표 2>에 CS 시스템의 실험에 사용된 기본 파라미터 값들을 정의해 놓았으며, 특별한 언급이 없는 경우 모든 실험에 이 값들을 사용하였다[20]. 파라미터 값들 중 재활작업시간($1/\mu_r$)과 작업전이 시간($1/\lambda_s$)은 10분으로 동일하게 처리하였으며, 이것은 두 과정 모두 컴퓨터의 부팅 소요 시간과 관련이 있기 때문이다. 모든 결함으로 인한 고장난 서버의 평균 수리시간($1/\mu$)은 12시간이 걸리며, 고장률(λ)은 3달에 한번 발생하고, 재활주기($1/\lambda_r$)는 2달에 한번 실행하는 것으로 가정하였다. 계획된 시스템의 중단 즉, 재활작업으로 인한 손실비용은 5이며 작업전이로 인한 손실비용은 30으로 하였으며, 이는 갑작스런 중단으로 인해서 발생하는

손실이 재활로 인한 손실보다는 크다는 것을 표시한다.

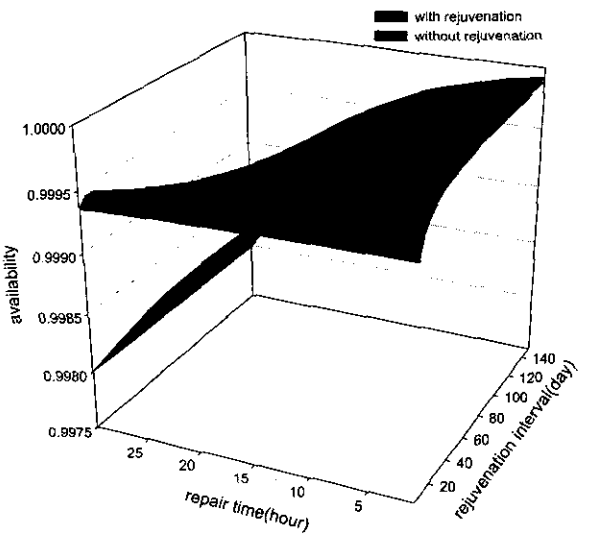
〈표 2〉 시스템의 기본 파라미터 값[20]

시스템의 기본 파라미터	값
서버의 수	2대
서버의 고장주기	90일
불완전한 상태 전이주기	7일
서버의 재활주기	60일
서버의 수리시간	12시간
재활작업 시간	10분
작업전이 시간	10분
단위시간당 고장발생 손실비용	1000
단위시간당 작업전이 시간 손실비용	30
단위시간당 재활작업 시간 손실비용	5
전체 시스템의 작동 시간	365일

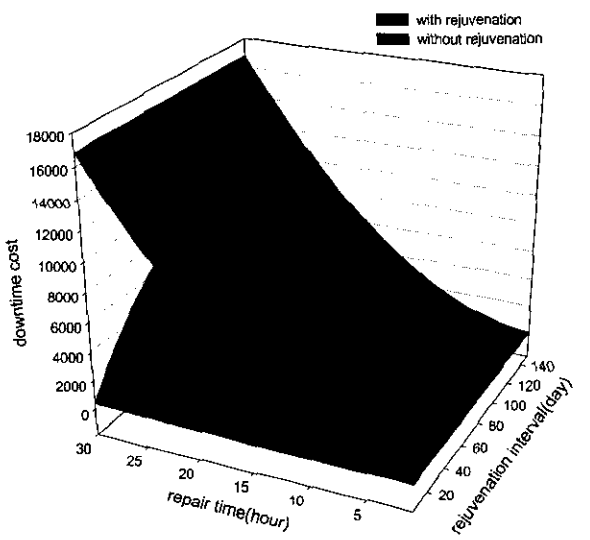
(그림 3)에 있는 상태 모델에 기초하여 재활주기에 대한 가용도와 손실비용의 변화 및 작업전이로 인한 가용도와 손실비용에 대한 분석과 작업전이가 시스템의 재활정책에 미치는 영향에 관한 실험을 하였다. 본 실험에서는 아래와 같은 5가지 질문에 대해서 중점적으로 살펴보았다.

- 작업전이 시간이 재활정책에 어떤 영향을 주는가?
- 작업전이 시간이 가용도에 어떤 영향을 주는가?
- 작업전이 시간이 손실비용에 어떤 영향을 주는가?
- 서버가 3대 이상일 경우에 재활정책이 2대일 때와 어떻게 달라지는가?
- 몇 대의 서버로 운영하는 것이 비용·효율적인 면에서 유리한가?

(그림 4)는 수리시간(repair time)과 재활주기(rejuvenation interval)에 대한 가용도와 손실비용의 변화를 보여주고 있다. (그림 4)의 (a)에서는 수리시간이 길어짐에 따라서 가용도가 떨어지는 것을 알 수 있는데, 이는 시스템을 처음 가동한 상태로 되돌리는데 필요한 시간이 늘어나기 때문에 그 만큼 전체 시스템의 다운 시간이 증가하고 가용도는 떨어지는 것이다. 재활주기에 따른 가용도 변화는 재활주기가 길어질수록 가용도가 항상 높아지는 것은 아니다. 이는 수리시간과 재활주기의 상관관계를 나타내는 것으로 수리시간이 길어지면 재활을 자주 함으로써 서버의 고장을 사전에 막아 주는 것이 바람직하며, 수리시간이 짧아지면 전체 시스템의 다운 시간이 감소할 것이므로 잦은 재활을 피하는 것이 좋다는 것을 의미한다. 손실비용 그래프에서는 가용도가 높으면 손실비용은 작게 나타났지만 수리 시간이 10시간 미만일 경우에는 가용도가 높음에도 불구하고 더 많은 손실비용을 초래하게 되는 것을 볼 수 있다. 이는 가용도가 절대적인 시간으로 표현했을 때의 값을 의미한다면, 손실비용은 글자그대로의 의미이외에, 가용도에 대한 다른 표현방법으로써 사용되었다고 할 수 있다. 즉, 사용자가 서비스 요청을 했을 때 요청된 서비스의 처리 유·무를 판단해서 서비스를 제공할 수 없는 상태에 대한 가용도를 비용적인 측면으로 나타낸 것이다. 그러므로 재활 주기를 결정할 때는 가용도만을 가지고 결정하는 것보다는 가용도와 손실비용을 모두 고려하는 것이 바람직하다.



(a) 가용도

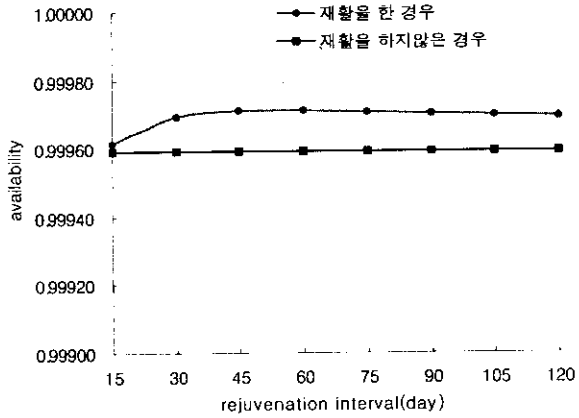


(b) 손실비용

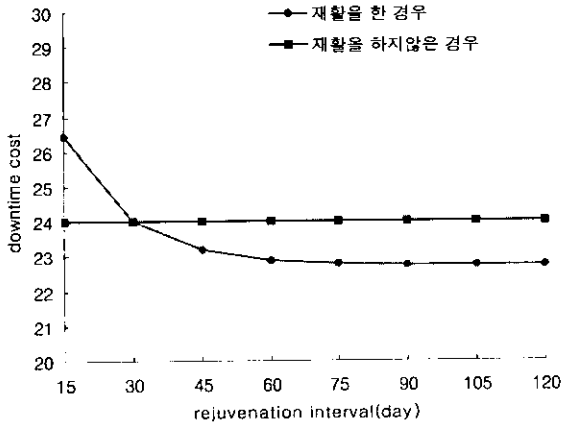
(그림 4) 수리시간과 재활주기에 대한 가용도와 손실비용의 변화

(그림 5)는 수리시간이 12시간일 때 가용도와 손실비용의 변화를 나타낸다. (그림 5)를 보면 가용도가 최대인 점에서 손실비용이 최소 값을 갖지 않기 때문에 재활 주기를 결정할 때 가용도가 최대인 점과 손실비용이 최소인 점에 근접한 값을 갖는 것으로 재활주기를 결정하는 것이 바람직하다. 따라서 가용도와 손실비용을 고려한 재활주기를 결정한다면 60~75일로 하는 것이 최적일 것이다. 이는 재활주기가 60~75일 사이에서 가용도와 손실비용 모두 최적의 값을 갖지는 않지만 두 값 모두 최적 값에 가장 근접하기 때

문이다. 이것은 하나의 예라고 할 수 있으며, 재활주기를 결정할 때는 가용도와 손실비용의 이해 득실을 고려해서 가장 효율적인 값으로 결정할 필요가 있다.



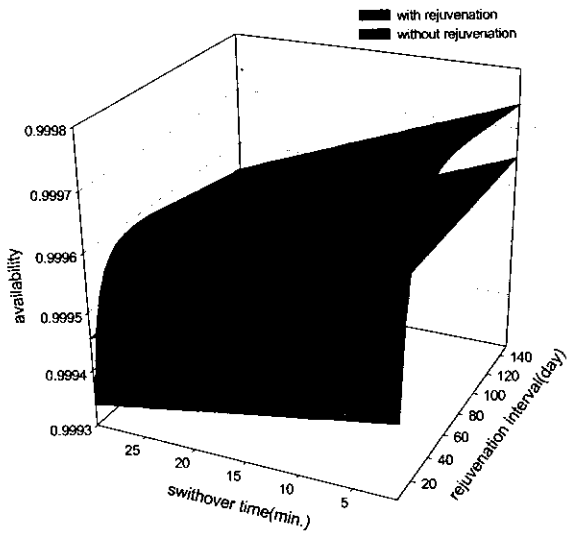
(a) 가용도



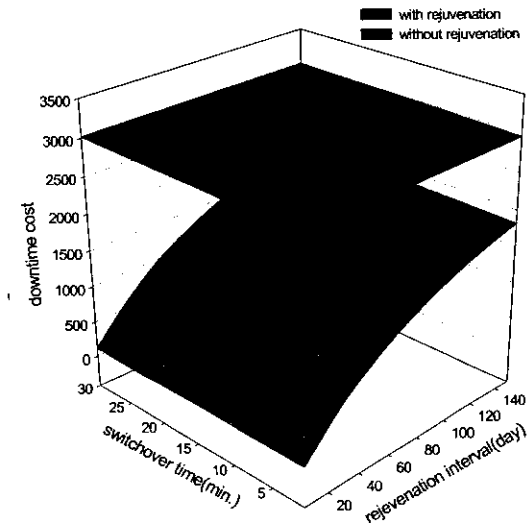
(b) 손실비용

(그림 5) 재활주기에 대한 가용도와 손실비용의 변화

(그림 6)은 작업전이 시간과 재활주기에 대한 가용도와 손실비용의 변화를 표시하고 있다. 가용도는 작업전이 시간이 짧을수록 증가하며, 재활주기에 대해서는 작업전이 시간에 따라서 위로 볼록한 모양을 하고 있다. 이는 작업전이 시간이 늘어날 경우, 재활주기가 길어짐에 따라 재활작업 수보다 작업전이 수가 더 많아지기 때문에 가용도가 어느 정점을 기준으로 떨어지는 현상을 보이는 것이다. (그림 6)의 (a)는 작업전이 시간이 모델링에 추가됨으로 인해서 [22]와는 정반대의 결과를 나타내고 있으며, 이는 CS 시스템에서 작업전이 시간이 재활정책 수립에 중요한 요소임을 의미한다. (그림 6)은 재활작업으로 인한 손실비용이 작업전이로 인한 손실비용보다 작기 때문에 재활을 자주 함으로써 CS 시스템의 손실비용을 줄일 수 있음을 보여주지만 이로 인한 가용도의 희생이 따름을 알 수 있다. 이는 재활작업 과정이 시스템의 가동 유무와 관련이 있기 때문이다.



(a) 가용도

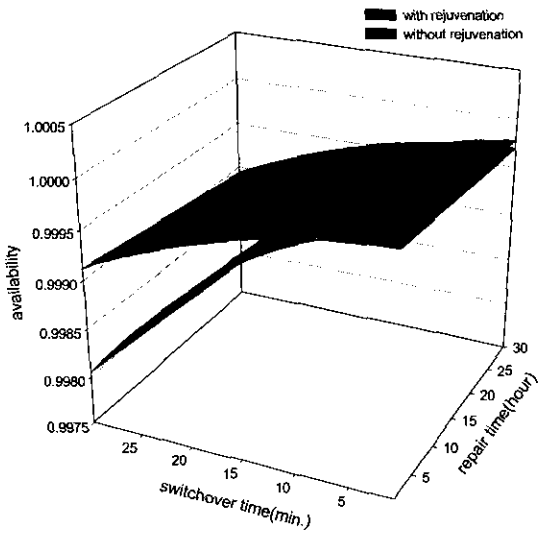


(b) 손실비용

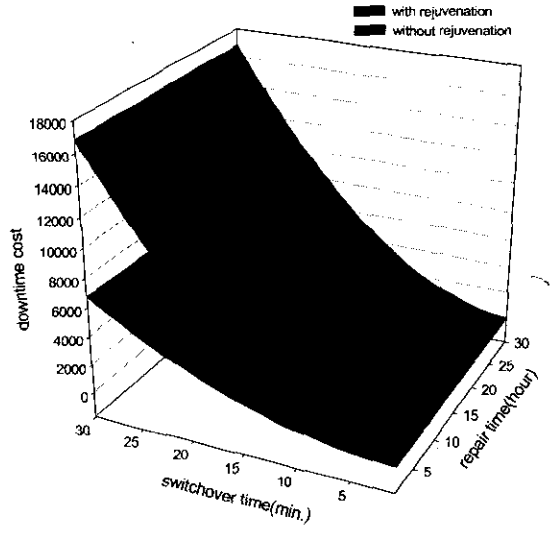
(그림 6) 작업전이 시간과 재활주기에 대한 가용도와 손실비용의 변화

(그림 7)은 작업전이 시간과 수리시간에 따른 가용도와 손실비용을 나타낸다. 작업전이 시간의 변화에 따른 가용도와 손실비용의 변화률이 수리시간에 대한 것보다 큼을 알 수 있다. 또한 작업전이 시간이 10분 미만일 경우에는 재활을 하지 않을 때 가용도가 높게 나타났지만 손실비용은 반대 현상을 보였다. 작업전이에 걸리는 시간이 10분이라는 것은 재활작업을 하는데 소요되는 시간과 같은 시간을 의미하지만 (그림 7)의 (b) 그래프가 보여주듯이 작업전이 시간이 재활작업 시간보다 더 적은 시간이 걸리더라도 손실비용은 재활을 하는 것이 좋다는 것을 알 수 있다. 이는 예방적 유지보수를 통해서 시스템을 좀더 비용·효율적으로 운영할 수 있다는 것

으로 소프트웨어 재활의 필요성을 제시하고 있다.



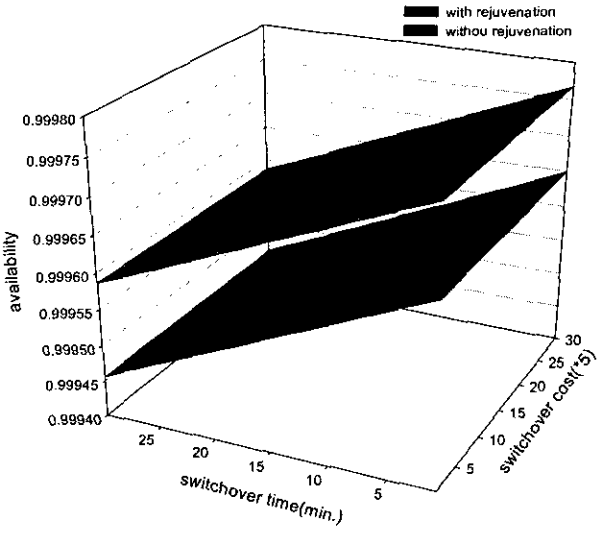
(a) 가용도



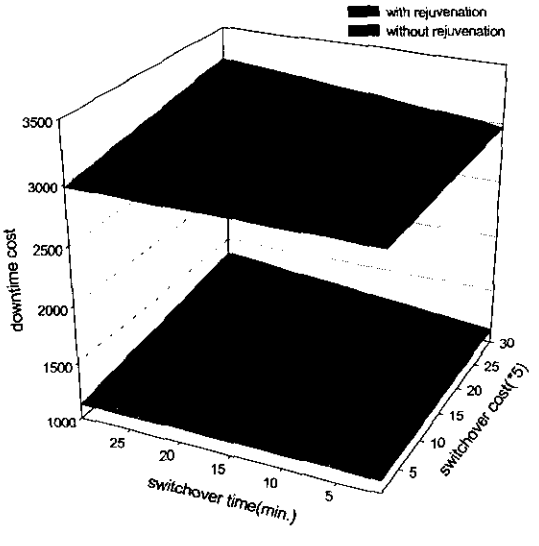
(b) 손실비용

(그림 7) 작업전이 시간과 수리시간에 따른 가용도와 손실비용의 변화

(그림 8)은 작업전이 시간과 작업전이로 인한 손실비용의 변화에 따른 가용도와 전체 시스템의 손실비용의 변화를 보여주고 있다. (그림 8)에서 작업전이 손실비용(switch-over cost)의 기본 단위는 5이며, 축의 값들은 배수를 의미한다. (그림 8)을 보면 가용도와 손실비용에 영향을 미치는 값들이 서로 다름을 알 수 있는데 이는 가용도가 시간적인 의미를 지닌 성능평가 척도이기 때문에 작업전이 시간에 의해 더 많은 영향을 받는 반면에, 손실비용은 시간적인 측면보다는 비용적으로 더 많은 영향을 받기 때문에 작업전이로 인한 손실비용에 따른 변화폭이 더 크게 나타나는 것이다.



(a) 가용도



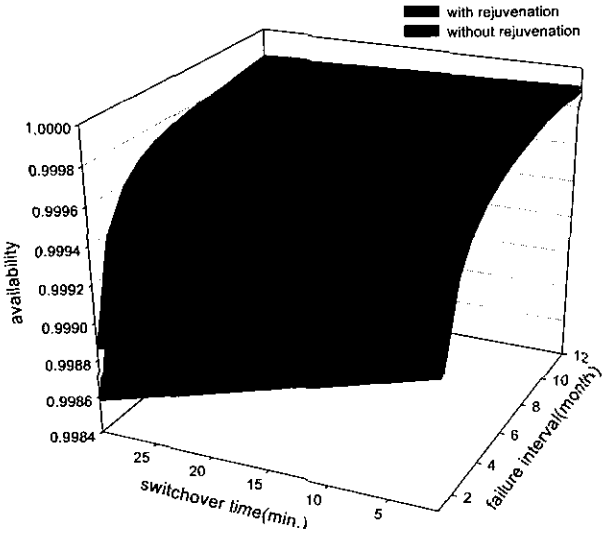
(b) 손실비용

(그림 8) 작업전이 시간과 작업전이로 인한 손실비용(C_s)의 변화에 대한 가용도와 손실비용의 변화

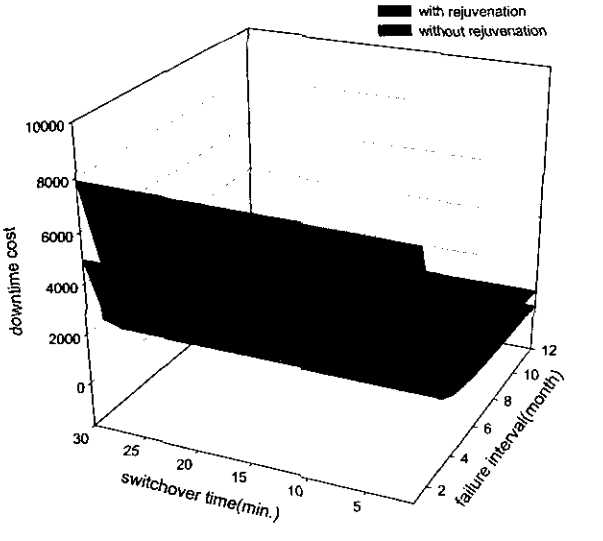
(그림 9)는 작업전이 시간과 고장주기에 따른 가용도와 손실비용의 변화 그래프이다. (그림 9)에서도 재활을 할 경우에 손실비용이 적음을 보여주고 있으며, 단지 가용도만이 고장주기가 10달이 넘을 경우에 역전되는 현상이 나타난다. 그렇지만 재활을 통한 가용도의 손실보다는 손실비용에서 얻는 이득이 상대적으로 많기 때문에 재활을 하는 것이 하지 않는 것보다는 좋음을 알 수 있다.

CS 시스템의 작업시간에 따른 변화를 보면 작업전이 시간과 가용도는 서로 상반관계를 보였다. 가용도를 보면, 재활을 했을 경우 항상 높은 값을 갖지는 않으며, 이것은 소프트웨어 재활이 미래에 닥쳐올 불시적인 결함을 예방하기 위해서 계획적으로 시스템의 가동을 멈추는 것이기 때문이다. 하지만 손실비용은 모든 경우에 재활을 하는 것이 유리

하게 나타났으며, 이것은 결함 발생을 방지해주는 기법인 소프트웨어 재활이 갑작스런 결함의 발생 회수를 줄여주고 손실 또한 적게 해주는 좋은 방법 중의 하나라는 것을 보여주는 것이라 할 수 있다.



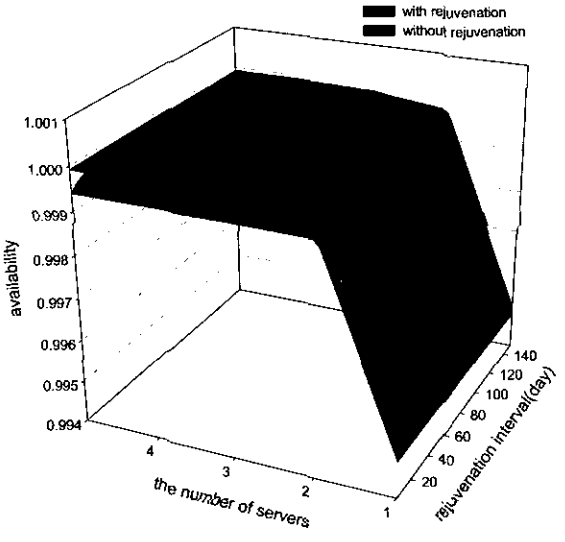
(a) 가용도



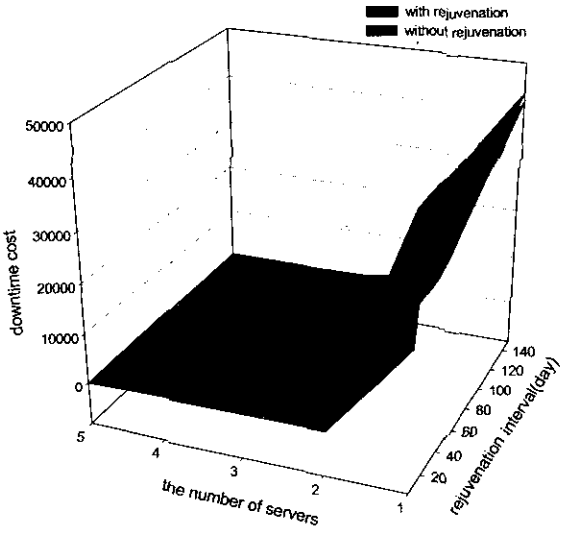
(b) 손실비용

(그림 9) 작업전이 시간과 고장주기에 대한 가용도와 손실비용의 변화

(그림 10)은 서버의 수와 재활주기에 대한 가용도와 손실비용의 변화를 나타낸다. 이 실험은 몇 대의 서버로 시스템을 운영하는 것이 유리한지를 알아보기 위한 것으로, 서버의 수가 3대 이상일 경우에는 서버 수의 증가로 인한 가용도의 증가폭과 손실비용의 감소폭이 서버를 이중계(서버 2대)로 구성했을 때보다 적었고, 재활주기에 따른 영향을 많이 받지 않았다. 따라서 한 대의 주서버와 한 대 또는 최대 2대의 여분서버를 가지고 CS 시스템을 구성하는 것이 비용 측면에서 유리함을 알 수 있다.



(a) 가용도



(b) 손실비용

(그림 10) 서버 수와 재활주기에 대한 가용도와 손실비용의 변화

5. 결 론

본 논문에서는 CS 시스템의 소프트웨어 재활 기법에 영향을 주는 중요한 파라미터인 작업전이 시간을 분석하였고, 재활 정책의 결정에 관해 논하였다. 비록 작업전이 시간이 고장주기나 다른 시스템의 파라미터 값들보다는 훨씬 작은 시간이기 때문에 간과하기 쉽지만, 작업전이 시간은 이중계로 이루어진 시스템에서는 재활의 유·무를 결정하는데 가장 중요한 요소임을 본 실험을 통해서 알 수 있었다. 재활주기를 결정하는데는 수리시간과 작업전이 시간이 많은 영향을 미치며, 가용도와 손실비용 모두를 고려해서 재활주기를 결정하

는 것이 바람직하다. 그 외에 서버의 고장주기, 서버의 불안정한 상태 전이 등은 재활정책과 재활주기에는 큰 영향을 주지 못했으며, 마지막으로 CS 시스템에서는 서버의 수를 2대 혹은 3대로 구성하는 것이 가장 비용·효율적임을 발견하였다.

향후 과제로는 시스템의 작업부하(workload)가 재활 정책에 어떤 영향을 주는지에 대한 연구가 필요하고, 소프트웨어의 재활로 인해서 서비스를 중지시켰을 때 현재 진행중인 서비스의 작업 완료시간을 줄이기 위해 checkpointing 기법을 이용해서 다시 서비스를 하는 것에 대한 연구가 필요하다.

참 고 문 헌

[1] 김석우, 서창호, "전자상거래 인증서비스 기술", 한국정보처리학회지, 제7권 제2호, pp.20-24, 2000.

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

[3] R. Jain, "The Art of Computer Systems Performance Analysis," pp.685, John Wiley & Sons Inc. 1991.

[4] J. Andreoli, F. Pacull and R. Pareschi, "XPECT : A Framework for Electronic Commerce," IEEE Internet Computing, pp.40-48, June, 1997.

[5] Z. Tian, L. Liu, J. Li, J. Chung and V. Guttemukkala, "Business-to-Business E-Commerce with Open Buying on the Internet," WECWIS, pp.56-62, April, 1999.

[6] 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.

[7] J. Gray and D. Siewiorek, "High-Availability Computer Systems," IEEE Computer, pp.39-48, September, 1991.

[8] M. Sullivan and R. Chillarege, "Software Defects and Their Impact on System Availability - A Study of Field Failures in Operating Systems," IEEE International Symposium on Fault-Tolerant Computing, pp.2-9, June, 1991.

[9] A. Pfening, S. Garg, A. Puliafito, M. Telek and K. Trivedi, "Optimal Rejuvenation for Tolerating Software Failures," 27th & 28th Performance Evaluation, pp.491-506, October, 1996.

[10] Y. Wang, Y. Huang, K. Vo, P. Chung and C. Kintala, "Checkpointing and Its Applications," Proceedings of 25th IEEE Fault-Tolerant Computing Symposium, pp.22-31, June, 1995.

[11] S. Garg, A. Moorsel, K. Vaidyanathan and K. Trivedi. "A Methodology for Detection and Estimation of Software Aging," Proceedings of 9th International Symposium on Software Reliability Engineering, pp.282-292, November, 1998.

[12] J. Gray, "Why Do Computers Stop and What Can Be Done About It?," Proceedings of 5th Symposium on Reliability in Distributed Software and Database Systems, pp.3-12, January, 1986.

[13] B. Grey, "Making SDI Software Reliable Through Fault-Tolerant Techniques," Defense Electronics, pp.77-80, 85-86, August, 1987.

[14] E. Marshall, "Fatal Error : How Patriot Overlooked a Scud," Science, pp.1347, March, 1992.

[15] A. Tai, S. Chau, L. Alkalaj and H. Hecht, "On-Board Preventive Maintenance : Analysis of Effectiveness and Optimal Duty Period," Proceedings of 3rd International Workshop on Object-Oriented Real-time Dependable Systems, pp.26-27, February, 1997.

[16] S. Garg, Y. Huang, C. Kintala and K. Trivedi, "Time and Load Based Software Rejuvenation : Policy, Evaluation and Optimality," Proceedings of the First Conference on Fault Tolerant Systems, pp.22-25, December, 1995.

[17] S. Garg, A. Puliafito, M. Telek and K. Trivedi, "Analysis of Software Rejuvenation Using Markov Regenerative Stochastic Petri Net," Proceedings of the Sixth International Symposium on Software Reliability Engineering, pp.180-187, October, 1995.

[18] S. Garg, A. Puliafito, M. Telek and K. Trivedi, "On the Analysis of Software Rejuvenation Policies," Annual Conference on Computer Assurance(COMPASS), pp.16-20, June, 1997.

[19] 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, January, 1998.

[20] Y. Hung, 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), pp.381-390, June, 1995.

[21] S. Garg, Y. Huang, C. Kintala and K. Trivedi, "Minimizing Completion Time of a Program by Checkpointing and Rejuvenation," ACM SIGMETRICS Conference, pp.252-261, May, 1996.

[22] 박기진, 김성수, 김재훈, "소프트웨어 재활 기법을 적용한 다중계 시스템의 가용도 분석," 한국정보과학회논문지(시스템및이론), 제27권 제8호, pp.730-740, 2000.

[23] 박기진, 김성수, "고가용도 Cold Standby 클러스터 시스템 성능 분석," 한국정보과학회논문지(시스템및이론), 제28권 제3·4호, pp.173-180, 2001.

이 재 성

e-mail : kidms@madang.ajou.ac.kr

1999년 아주대학교 정보및컴퓨터공학부 (공학사)

2000년~현재 아주대학교 정보통신전문대학원 석사과정

관심분야 : 클러스터 시스템, 성능분석, 결함허용

박 기 진

e-mail : kiejin@etri.re.kr

1989년 한양대학교 산업공학과(공학사)

1991년 포항공과대학교 산업공학과 (공학석사)

2001년 아주대학교 컴퓨터공학과(공학박사)

1991년~1996년 삼성종합기술원 기반기술 연구소 전임연구원

1996년~1997년 삼성전자(주) 소프트웨어센터 선임연구원

2001년~현재 한국전자통신연구원(ETRI) 네트워크장비시험 센터 선임연구원

관심분야 : 통신 시스템, 고가용성 시스템, 결함허용 시스템, 성능 평가, 시뮬레이션, 신뢰도

김 성 수

e-mail : sskim@madang.ajou.ac.kr

1982년 서강대학교 전자공학과(공학사)

1984년 서강대학교 전자공학과(공학석사)

1995년 Texas A&M University, Computer Science Dept.(공학박사)

1983년~1986년 삼성전자(주) 종합연구소 컴퓨터연구실(주임연구원)

1986년~1996년 삼성종합기술원(수석연구원)

1991년~1992년 Texas Transportation Institute(연구원)

1993년~1995년 Texas A&M University, Computer Science Dept.(T.A. & R.A.)

1997년~1998년 한국정보처리학회, 한국정보과학회 논문지 편집위원

1996년~현재 아주대학교 정보및컴퓨터공학부/정보통신 전문대학원 부교수

관심분야 : 고가용성 시스템, 결함허용, 성능 평가, 이동컴퓨팅, 멀티미디어 등