소프트웨어 재활 기법을 적용한 고가용성 시스템의 설계

박기진°, 김성수
아주대학교 정보통신대학 정보및컴퓨터공학부

A Design of Highly Available Systems using Software Rejuvenation Method

Kiejin Park, Sungsoo Kim
Division of Information and Computer Engineering, Ajou University

요 약

고가용성 시스템의 소프트웨어 재활 기법은 시스템의 결함 발생 이후에 수동적으로 대처하기보다는 결함이 발생하기 전에 이를 미연에 방지하는 능동적 차원의 결함 허용 방법이다. 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.

1. 서론

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 타스크 상태를 저장하여 소프트웨어 결함시 가장 최근의 타스크 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 데이터를 이중계로 구성하여 타스크를 수행하는 방법은 자원의 낭비가 심하고 복사된 프로그램 또는 데이터들의 일치성을 유지시키는데 많은 어려움이 있다. 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, 항공 산업 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, 데이터 유실 등으로 인한 노화 현상이 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.

Trivedi[1]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. Huang[2,3]은 유지 보수 기간동안에 서버가 가동하지 않아서 발생하는 비용 증가를 고려한 최적 원상 복구 작업 조건을 구하였으나, state transition model 이 단순하며, 유지보수 주기를 임의로 선정하고 있다. 또한 작업 완료 시간을 최소화하기 위해, checkpointing 기법과 재활 기법을 결합하여 연구한 사례도 있으며, 최근 Motorola에서는 무선기기의 고장 및 에러를 관리하기 위해, 기존의 유선 라인에서 사용되었던 하드웨어 이중화 기법 대신에, 소프트웨어 기법을 활용한 이동 컴퓨팅 환경의 서버와 정보 단말기간의 동적 데이터에 대한 재활 전략을 연구 중에 있다.

멀티미디어 이동 컴퓨팅의 특성상 무정지 요건을 갖추기 위해서는 고수준의 가용성을 보장해 주는 다중계 시스템 구성이 일반적이지만

본 연구는 1999년도 정보통신부 정보통신 우수시범학교 지원사업에 의한 결과임.

(예를 들어 이중계로 구성된 시스템에서는 두 대의 서버가 동시에 가동되며, 만약 어느 한 서버에 이상이 발생할 경우, 즉각 다른 서버에서 업무를 처리하게 된다[4].) 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, n개의 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) 시스템에서의 소프트웨어 재활에 관한 주제를 다루었다.

3. 시스템 모델

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, 주 서버를 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 주 서버가 재활 작업 대상이며, 가동되고 있지 않는 서버는 고려 대상에서 제외된다. 재활 작업 시간($1/\mu_r$)은 서버수에 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)

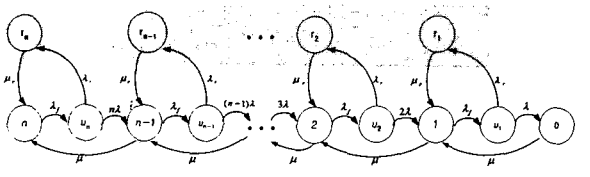


그림 1 소프트웨어 재활을 고려한 HSS 시스템 상태 모델

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 변화률로 재활 작업에 들어가거나, 혹은 $i*\lambda$의 변화률(HSS 시스템의 경우 고장이 나지 않은 i 대의 서버가 동시에 가동함)로 고장이 발생하게 된다. 또한 normal state에서 unstable state로의 변화률은 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)

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

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

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

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

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

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.

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

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

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

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

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

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 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.

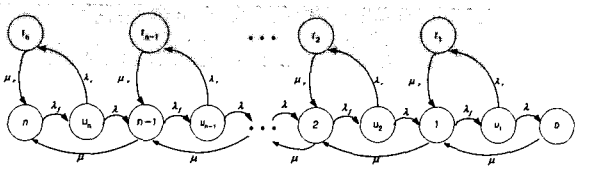


그림 2 소프트웨어 재활을 고려한 CSS 시스템 상태 모델

그림 2의 평형상태에서의 확률은 다음과 같다.

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

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

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

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

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_{r1})할 때를 제외한 경우를 고려함으로써 계산될 수 있다.

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

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

CSS 시스템에서 재활을 고려한 경우의 가용도 계산은 다음과 같다.

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

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

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.($C_f >> C_r$)

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

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

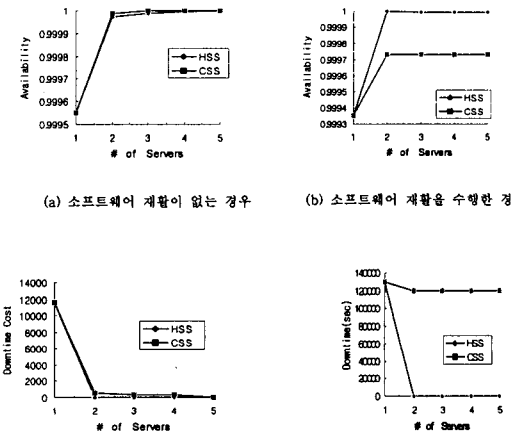
멀티미디어 이동 서비스의 특성상 가용도가 특히 중요시되는 컴퓨팅 시스템에 적합한 재활 정책을 개발하기 위하여, 가동 시간, 가동 방법, 고장률, 수리률 및 발생 비용 등의 변수를 복합적으로 고려하였다. downtime의 길이에 대한 고려보다는 이로 인해 발생하는 손실 비용이 더욱 중요한 요소가 되기 때문에 다양한 시스템 운영 파라미터를 동시에 고려하여 가용도 및 손실 비용을 계산하였다. HSS 혹은 CSS 방식으로 가동되는 서버 2대의 가동기간(T)은 1년으로 하며, 서버의 고장(λ)은 1년에 1회, 고장 수리(μ)에 1시간이 소요되고,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

그림 3은 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 4 (a) 참조). 하지만 downtime으로 인해 발생하는 비용은 두 시스템 모두 재활을 많이 하면 할수록 감소하는 추세를 보였으며(그림 4 (b) 참조), CSS 시스템의 경우 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.

5. 결론

본 연구에서는 HSS와 CSS 시스템의 상태 전이 모델의 평형 상태에서의 확률을 구하여, 시스템이 가동중이거나, 혹은 재활 작업 또는 고장 상태에 있을 확률을 계산하였다. 평형 상태에서의 가용도 계산을 통해, 일정 기간 동안의 시스템 downtime과 이로 인해 발생하는 손실 비용 함수를 정의하였고, 수학적 분석을 통하여 구한 재활 모델의 closed-form 해를 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.

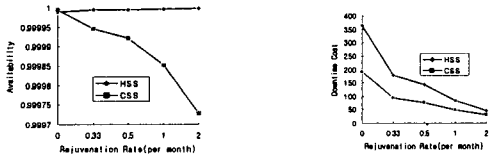
소프트웨어 최적 재활 주기를 산정할 때에, 시스템의 부하를 고려해서 정책을 수립할 필요성이 있고 checkpointing방법을 소프트웨어 재활 기법과 결합시킬 경우에 평균 완료시간(expected completion time)을 더욱 더 감소시킬 수 있을 것으로 기대된다.



(a) 소프트웨어 재활이 없는 경우 (b) 소프트웨어 재활을 수행한 경우

(c) 손실 비용 (d) Downtime

그림 3 중복도(n)에 따른 가용도, 손실 비용 및 Downtime의 변화



(a) 가용도 (b) 손실 비용

그림 4 재활률(λ_r)에 대한 가용도 및 손실 비용의 변화

참고문헌

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

[2] Y. Huang, C. Kintala, N. Kolettis and N.D. 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.

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

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