2000년 한국정보처리학회 추계 학술발표논문집 제7권 제2호

작업전이 시간을 고려한 주-백업 시스템의 성능평가

이재성*, 박기진**, 김성수*
*아주대학교 정보통신전문대학원
**아주대학교 대학원 컴퓨터공학과
e-mail:{kidms, sskim}@madang.ajou.ac.kr

Performance Evaluation of Primary-Backup System Regarding Switchover Time

Chaesung Lee*, Kiejin Park**, Sungsoo Kim*
*Professional Graduate School of Information and Communication Technology, Ajou University
**Department of Computer Engineering, Graduate School of Ajou University

요 약

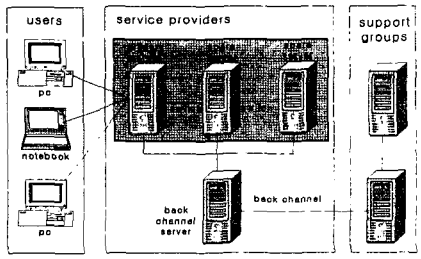
인터넷의 보급 확산이 증가됨에 따라 컴퓨터 시스템은 점차 복잡하고 규모가 커지기 때문에 시스템 장애로 인한 피해도 증가하고 있다. 컴퓨터 시스템의 신뢰도가 중요시되는 가운데 오늘날은 하드웨어보다 소프트웨어의 신뢰도가 더욱 중요시되고 있으며, 소프트웨어의 신뢰도를 향상시키기 위해서는 소프트웨어로 인한 결함 발생을 줄여주는 것이 중요하다. 본 연구에서는 작업전이 시간을 고려한 주-백업 시스템에서의 모델링을 제안하였으며 가용도와 손실비용에 대한 분석을 하였다. 분석을 통해서 작업전이 시간이 주-백업 서버 시스템에서의 재활정책에 가장 중요한 요소임을 발견하였다.

1. 서론

인터넷의 급속한 성장과 인터넷 이용자의 급격한 증가로 인해서 컴퓨터 시스템은 점차 대규모화·복잡화되고 있다. 현대 사회에서는 컴퓨터 시스템에 대한 의존도가 점점 더 높아지고 있으므로 컴퓨터 시스템의 결함으로 인한 피해는 상상할 수 없을 정도로 크며 앞으로는 컴퓨터 시스템의 가용도가 더욱 더 중요시 될 것으로 예측된다[1,2].

하드웨어적으로 가용도를 높이기 위한 방법으로 하나의 주 서버(primary server)를 대체하는 여분 서버(spare server)를 두는 방법이 사용되며, 이것은 주 서버에서 여분 서버로의 작업전이 방법에 따라서 CS(cold standby)방법과 HS(hot standby)방법으로 구분할 수 있다. 위의 두 가지 방법 중에서 HS 시스템은 주 서버를 대체하는 시간이 매우 짧지만 모든 서버를 동시에 돌려야 하므로 CS 시스템에 비해서 비용이 많이 들고 서버의 결함 발생률이 상대적으로 큰 반면에, CS 시스템은 HS 시스템보다 주 서버를 대체하는 시간은 길지만 비용이 적게 들고 서버의 결함 발생률이 상대적으로 적기 때문에, 높은 수준의 가용도를 요구하는 시스템에서는 HS 시스템

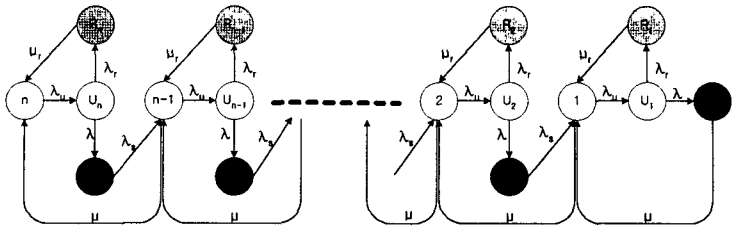
This work is supported in part by the Ministry of Information & Communication of Korea("Support Project of University Foundation Research<'00>" supervised by IITA)



(그림 1) 다중 서버로 구성된 CS 시스템

을 사용하는 것이 유리하고 상대적으로 낮은 수준의 가용도를 요구하는 시스템에서는 CS 시스템을 사용하는 것이 바람직하다. 또한 높은 수준의 가용도를 요구하지만 비용적인 측면을 고려해야 한다면 일반적으로 CS 시스템을 사용하는 것이 유리하다.

(그림 1)은 다중 서버로 구성된 CS 시스템의 구성 예이다. (그림 1)을 보면 사용자는 인터넷을 통해 서비스 제공자의 주 서버에 접근을 해서 본인들이 원하는 정보를 요청하고 서비스를 제공받는다. 이때 주 서버에서 모든 사용자에게 요청된 서비스를 제공하는 도중 결함이 발생하면 여분 서버가 주 서버를 대신해서 서비스를 제공하며, 거래에 필요한 여러 가지 다른 정보들은 다른 서비스 제공자의 시스템과 연결되어 있는 back channel 서버와의 통신



(그림 2) 작업전이 시간을 고려한 CS 시스템의 상태 모델

을 통해서 얻는 방식으로 작동한다.

본 연구에서는 CS 시스템에서의 가용도를 높이고 손실비용을 줄이기 위한 방법으로 소프트웨어 재활 기법을 사용하였다. 최근에는 기술의 발전으로 하드웨어에 의한 결함 발생률은 점차로 줄어들고 있으며, 오히려 소프트웨어에 의한 결함이 전체 시스템 결함의 62%를 차지하는 것으로 보고되었다[3]. 결함 발생 후 결함을 복구하는 기존의 방법과 달리 소프트웨어 재활은 일시적인 결함(transient fault)에 기인한 소프트웨어 결함이 발생하기 전에 결함의 원인을 없애주는 방법으로 시스템의 운영자나 사용자의 입장에서 보면 시스템을 효율적으로 운영할 수 있게 해주는 좋은 방법이라고 판단된다. 본 연구에서는 (그림 1)의 회색영역의 서버를 대상으로 가동중인 주 서버와 주 서버에 결함이 발생했을 때만 가동되는 다수의 여분 서버들로 구성된 CS 시스템에서, 소프트웨어 재활 기법을 통한 서버의 가용도와 손실비용을 줄이기 위한 재활 정책을 연구하였다.

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

2. 관련 연구

소프트웨어는 오랜 기간동안 계속해서 실행을 하면 메모리 부족, 데이터의 불일치, 저장 공간의 단편화 현상과 같은 원인으로 인한 노화현상이 생기며, 결국은 소프트웨어의 상태를 불안정한 상태로 동작시킴으로써 컴퓨터 시스템의 결함을 발생시킨다. 컴퓨터 시스템의 결함을 발생시키는 원인이 되는 노화현상을 제거함으로써 결함을 사전에 막아보려는 방법이 소프트웨어 재활이며, 이것은 현재 서버에서 실행중인 소프트웨어가 불안정한 상태에서 실행을 하고 있을 경우 이를 처음 실행한 상태로 환원시켜 주는 것으로, 노화현상을 막고 노화로 인한 컴퓨터 시스템의 결함을 줄일 수 있게 해 준다[4,5].

서버가 1대로 구성된 시스템[6]과 서버가 2대 이상으로 구성된 시스템[1]에 대하여 소프트웨어 재활 기법을 적용한 기존 연구 사례가 있으며, 서버가 2대 이상으로 구성된 기존 연구에서는 CS 시스템에서의 재활에 대한 재활 정책에 관한 논의가 있었지만, 서버의 작업전이 시간을 고려하지 않았다. CS 시스템에서 작업전이에 소요되는 시간은 여분 서버를 켜고 서비스를 다시 시작 할 때까지의 시간으로 재활시간과 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. 소프트웨어 재활을 하는데 걸리는 시간은 최악의 경우 서버를 끄고 다시 켠다고 하더라도 작업전이 시간이하가 걸린다. 본 연구에서는 CS 시스템에서 서버의 작업전이 시간을 고려한 모델을 제안하고, 소프트웨어 재활을 했을 경우와 하지 않았을 경우에 대해서 손실비용 및 재활주기에 대해서 분석하였다.

3. CS 시스템 모델

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

- 재활작업 시간($1/\mu_r$)는 동일하다.
- 고장 난 서버를 수리하는 수리시간($1/\mu$)은 동일하다.
- n개의 서버로 구성된 시스템에서 각 서버의 고장률(λ)은 동일하다.
- 여분의 서버가 고장 난 서버를 대체하는 데 걸리는 작업전이 시간($1/\lambda_s$)은 동일하다.
- 재활을 할 때, 현재 가동 중인 서버만이 재활 대상이고 가동되지않는 여분의 서버는 재활 대상에서 제외된다.
- 서버의 가동을 주기적으로 멈추는 재활률(λ_r)은 모든 가동 상태에서 동일하다.
- 모든 상태에 머무는 시간은 지수분포를 따른다.

(그림 2)의 CS 시스템은 n대의 서버로 구성되어 있고 이 중 한 대를 제외한 n-1대의 서버가 여분 서버로 구성되어 있다. {n, n-1, ... , 1, 0}은 결함발

생이 적은 상태에서 가동되는 서버의 수이며 {0}은 시스템이 가동되지 않는 고장 상태를 가리킨다. $\{U_n, U_{n-1}, \ldots, U_2, U_1\}$은 서버의 장시간 가동으로 인한 서버의 불안정 상태를 나타내며, $\{R_n, R_{n-1}, \ldots, R_2, R_1\}$은 서버의 가동을 고의로 멈춘 후, 건강한 상태로 만드는 재활 상태를 의미한다. $\{S_{n-1}, S_{n-2}, \ldots, S_2, S_1\}$은 재활작업이나 고장으로 인한 서버간의 작업전이를 표시한다. 처음에 시스템은 i 상태에 있다가 서버가 계속해서 가동됨으로 인해서 U_i 상태로 바뀌며, U_i 상태에서는 서버의 결함이 발생하기 전에 재활을 해주는 재활 상태(R_i) 혹은, 결함 발생 후에 작업전이 상태(S_i)로 이동할 수 있다. 재활 상태에서는 재활작업 시간($1/\mu_r$) 후 결함 발생이 적은 i 상태에서 서버가 다시 가동되며, 작업전이 상태에서는 일정 작업전이 시간($1/\lambda_r$) 후 i-1 상태로 전이한다. 서버의 결함으로 i 상태에서 i-1 상태로 전이 후 일정 수리 시간($1/\mu$)후에 다시 i 상태로 이동한다.

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

$$\lambda_u \cdot P_n = \mu_r \cdot P_{r_n} + \mu \cdot P_{n-1}$$
$$(\lambda_u + \mu) \cdot P_{n-i} = \mu_r \cdot P_{r_{n-i}} + \lambda_r \cdot P_{s_{n-i}} + \mu \cdot P_{n-i-1} \quad ,i = 1, 2, \ldots, n-1$$
$$(\lambda + \lambda_r) \cdot P_{s_{n-i}} = \lambda_u \cdot P_{n-i} \quad ,i = 0, 1, \ldots, n-1$$
$$\mu_r \cdot P_{r_{n-i}} = \lambda_r \cdot P_{u_{n-i}} \quad ,i = 0, 1, \ldots, n-1$$
$$\lambda_s \cdot P_{s_{n-i}} = \lambda \cdot P_{u_{n-i+1}} \quad ,i = 1, 2, \ldots, n-1$$
$$\mu \cdot P_0 = \lambda \cdot P_{u_1}$$
$$\sum_{i=0}^{n} P_i + \sum_{i=1}^{n} (P_{u_i} + P_{r_i}) + \sum_{i=1}^{n-1} P_{s_i} = 1$$

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

$$P_{u_i} = \frac{\lambda_u}{\lambda + \lambda_r} \cdot P_i \quad ,i = 1, 2, \ldots, n$$
$$P_{r_i} = \frac{\lambda_r}{\mu_r} \cdot \frac{\lambda_u}{\lambda + \lambda_r} \cdot P_i \quad ,i = 1, 2, \ldots, n$$
$$P_{s_i} = \frac{\mu}{\lambda_s} \cdot P_i \quad ,i = 1, 2, \ldots, n-1$$
$$P_i = \left(\frac{\lambda \cdot \lambda_u}{\mu \cdot (\lambda + \lambda_r)} \right)^{n-i} \cdot P_n \quad ,i = 0, 1, \ldots, n$$
$$P_n = \left[\sum_{i=1}^{n} \left(\frac{\lambda \cdot \lambda_u}{\mu \cdot (\lambda + \lambda_r)} \right)^{n-i} \cdot \left\{ 1 + \frac{\lambda_u}{\lambda + \lambda_r} \cdot \left(1 + \frac{\lambda_r}{\mu_r} \right) + \frac{\mu}{\lambda_s} \right\} + \left(\frac{\lambda \cdot \lambda_u}{\mu \cdot (\lambda + \lambda_r)} \right)^{n} - \frac{\mu}{\lambda_s} \right]^{-1}$$

4. CS 시스템 모델 분석

모델 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 주 서버에 결함이 발생했을 때 여분 서버로의 작업전이 상태(P_{s_i}), 서버의 재활작업을 수행하는 상태(P_{r_i})를 제외시킴으로써 구할 수 있다. 재활을 하지 않았을 경우에는 재활이 필요 없으므로 가용도 식에서 P_{r_i}가 제외된다.

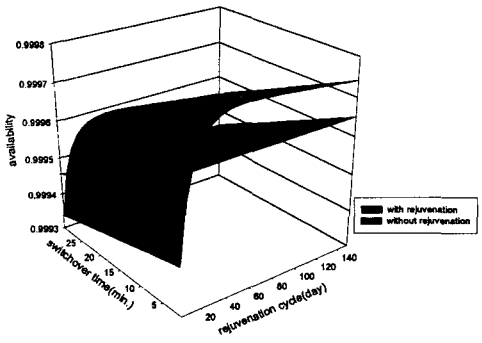
CS 시스템의 손실비용(downtime cost)은 서버의 결함으로 인해서 발생하는 손실로, 서비스를 제공받을 수 없는 상태인 P_{r_i}, P_{s_i}, P_0에 각각의 단위 시간당 손실비용인 C_r, C_s, C_f를 곱해서 단위 시간당 총 손실비용을 구한 후, 이를 전체 시스템의 가동시간(L)과 곱함으로써 계산된다. 특히, C_r, C_s, C_f는 서버의 결함으로 발생하는 단위 손실비용으로 C_r은 계획된 서버의 중지로 인해 발생하는 손실로 가장 적으며, C_s는 갑작스런 서버의 중지로 발생하는 손실로 C_r보다는 높으며 C_f 보다는 작고, C_f는 갑작스런 서버의 결함으로 전체 시스템이 다운되었을 때 발생하는 손실로 매우 높다. C_s가 C_r 보다 높은 이유는 시스템이 S_i 상태로 전이가 일어나는 시점(주 서버의 고장으로 여분 서버로의 작업전이가 일어나는 시점)을 예측할 수 없기 때문이다.

<표 1>에 CS 시스템의 실험에 사용된 기본 파라미터 값들을 정의해 놓았으며, 특별한 언급이 없는 경우 모든 실험에 이 값들을 사용하였다[6].

<표 1> 시스템의 기본 파라미터 값

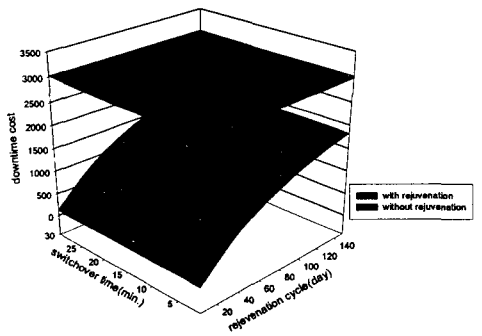
시스템의 기본 파라미터	값
서버의 개수	2대
서버의 고장주기	90일
불완전한 상태 전이주기	7일
서버의 수리시간	12시간
재활 시간	10분
작업전이 시간	10분
단위시간당 고장발생 손실비용	1000
단위시간당 작업전이 시간 손실비용	30
단위시간당 재활시간 손실비용	5
전체 시스템의 작동 시간	365일

본 논문에서는 작업전이 시간이 CS 시스템에 미치는 영향을 알아보기위해 가용도와 손실비용에 대한 분석을 하였다



(그림 3) 작업전이 시간과 재활주기에 대한 가용도의 변화

(그림 3)과 (그림 4)는 작업전이 시간과 재활주기에 대한 가용도와 손실비용의 변화를 표시하고 있다. (그림 3)에서 가용도는 작업전이 시간이 짧아질수록 가용도가 증가하며, 재활주기에 대해서는 작업전이 시간에 따라서 위로 볼록한 모양을 하고 있다. 이는 작업전이 시간이 증가할 경우, 재활주기가 길어짐에 따라 재활작업 횟수보다 작업전이 횟수가 더 많이 발생하기 때문에 가용도가 어느 정점을 기준으로 떨어지는 현상을 보인다. (그림 3)은 작업전이 시간이 모델링에 추가됨으로 인해서 [1]과는 정반대의 결과를 보여주고 있으며, 이는 CS 시스템에서는 작업전이 시간이 재활정책 수립에 중요한 역할을 한다는 것을 의미한다.



(그림 4) 작업전이 시간과 재활주기에 대한 손실비용의 변화

(그림 4)에서는 재활주기가 증가하면서 손실비용 또한 증가하는 것을 볼 수 있는데, 이는 재활작업으로 인한 손실이 작업전이로 인한 손실보다 작으므로 재활을 자주 해줌으로써 작업전이의 빈도 수가 적어지기 때문이며, (그림 3)에서 가용도가 증가함에도 불구하고 손실비용이 증가하는 것도 이를 나타낸다고 할 수 있다. 즉, 잦은 재활작업으로 인해서 가용도는 감소하지만 재활작업으로 서버의 고장 발생을 적게 해 줌으로써 작업전이 횟수가 적어짐으로써 전체적인 손실이 감소하는 것을 보여준다.

5. 결론

본 논문에서는 기존 연구에서 고려하지 않았던 작업전이 시간을 모델링에 포함시킨 소프트웨어 재활기법을 연구하였다. 작업전이 시간은 재활정책을 결정하는데 중요한 요소이며, 특히 CS 시스템에서의 분석 시에는 작업전이 시간을 반드시 고려해야 하는 것을 파악하였다.

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

참고문헌

[1] 박기진, 김성수, 김재훈, "소프트웨어 재활 기법을 적용한 다중계 시스템의 가용도 분석," 한국정보과학회논문지(시스템및이론), 제27권, 제8호, 2000.

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

[3] 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.

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

[5] 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, Jan. 1998.

[6] Y. Hung, C. Kintala, N. Kolettis and N. Fulton, "Software Rejuvenation: Analysis, Module and Applications," Proceedigns of the 25th International Symposium on Fault Tolerant Computing(FTCS-25), pp. 381-390, June 1995.