論文
3-1-9

소프트웨어 재활 기법을 사용한 클러스터 웹서버 시스템의 가용도 분석

(Availability Analysis of Cluster Web Server System using Software Rejuvenation Method)

강 창 훈*
(Chang-Hoon Kang)

요 약

다수의 서버를 클러스터로 연결하여 동시에 가동 할 경우 서버의 대수가 증가함으로 인해 발생하는 가용도 저하문제와 소프트웨어의 노화로 인하여 높은 가용도를 제공하기가 어렵다. 본 연구에서는 n대의 주 서버와 k대의 여분서버로 구성되는 클러스터 웹 서버 시스템에서 가동되는 서버의 수, 여분서버의 수, 소프트웨어의 재활주기, 재활소요시간, 서버의 고장률, 서버의 수리율, 서버의 불안정률 등의 시스템 운영 파라미터에 기초하여, 소프트웨어의 재활정책에 대한 평가를 위해 평형 상태에서의 확률, 정지시간, 가용도, 손실비용 등을 계산하였다. 수학적 분석을 통해 다양한 시스템 운영 상태에 대한 실험을 통해 검증하였으며, 소프트웨어의 재활 정책에 의한 예방적 결함허용 기법이 시스템의 안정성에 중요한 요소임을 확인하였고, 또한 서버의 고장률 및 불안정률이 소프트웨어 재활 정책 결정에 중요한 요소임을 파악하였다.

ABSTRACT

An cluster system used consist of large number of running servers, one has the problem that does the low availability occured by the high chance of the server failures and it is difficult to provide occuring software aging. In this paper, running cluster web servers consists of n primary servers and k backup servers, based on the operational parameters such as number of running primary servers, number of backup servers, rejuvenation period, rejuvenation time, failure rate of servers, repair rate of servers, unstable rate of servers. We calculate to evaluate the rejuvenation policy such steady-state probabilities, downtime, availability, and downtime cost. We validate the solutions of mathematical model by experiments based on various operation parameters and find that the software rejuvenation method can be adopted as prventive fault tolerant technique for stability of system. The failure rate and unstable rate of the servers are essential factors for decision making of the rejuvenation policies.

* 정회원 : 극동정보대학 멀티미디어과 조교수

논문접수 : 2002. 1. 12.
심사완료 : 2002. 1. 24.

1. 서론

최근 인터넷의 급속한 보급으로 인터넷을 활용한 사회 활동이 급증하면서 웹서버, 전자상거래서버, VOD 서버, 게임서버 등을 포함한 여러 분야에서 고성능 서버에 대한 수요가 증가하고 있다. 그러나 고성능 서버는 각 서버 회사의 고유한 아키텍쳐에 기반하여 캐비넷 형태로 생산됨으로 가격이 수억에서 수십 억에 이르고 있어 구입 및 활용에 상당한 어려움을 가지고 있다.

이에 대한 대안으로 클러스터 시스템이 대두되었다. 클러스터 시스템은 저가의 고성능 PC 및 워크스테이션들을 고속 네트워크를 사용하여 연결함으로써 고성능 서버의 성능을 발휘하는 시스템이다. 기존의 고성능 서버보다 아주 적은 비용으로 동일한 성능을 낼 수 있어 가격대 성능비가 높은 시스템이다[1,3,4,12].

그러나 다수의 서버를 연결함으로써 각 서버의 결함으로 인해 클러스터 시스템 전체적의 가용도(Availability)는 저하된다. 이에 따라 여분의 서버를 두어 고장난 서버를 신속하게 대치하는 방법이 연구되고 있다. 특히 사용자의 입장에서 가장 중요시되는 시스템 평가 척도는 사용자가 원하는 경우에 시스템에 접속해서 서비스를 받을 수 있는 지를 판단하는 가용도이며, 이는 평균고장시간(Mean Time To Failure : MTTF)만을 다루는 신뢰도와는 달리 평균수리 시간(Mean Time To Repair : MTTR)까지 포함하는, 즉 시스템이 고장났을 경우 이를 얼마나 신속하게 복구할 수 있는 능력을 고려한 측면에서 많은 연구가 이루어지고 있다[6].

한편 컴퓨터 시스템 기술의 발전과 사용자의 다양한 요구로 인하여 소프트웨어의 복잡도는 날로 증가되고 있는 추세이다. 이같이 복잡도가 증가됨에 따라 소프트웨어의 설계, 구현 또는 그 밖의 여러 가지 원인과 소프트웨어와 관련된 결함으로 인해 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 시스템 장애로 인한 피해는 이루 상상할 수 없을 정도로 크다[1,2,3]. 물론 컴퓨터 장애의 원인으로 하드웨어 결함에 의한 장애도 있지만, 소프트웨어에 의한 장애가 전체 장애의 62%를 차지하는 것으로 보고되었고[13], 앞으로 소프트웨어 결함에 의한 장애는 비중이 점차로 늘어날 전망이다. 특히 멀티미디어 이동 컴퓨팅이나 무정지 서비스를 해야 하는 인터넷 서버에서 사용되는 소프트웨어는 빈번한 통신 두절과 데이터 유실로 인해 소프트웨어의 결함이 더욱 심각할 가능성이 높다[5,9].

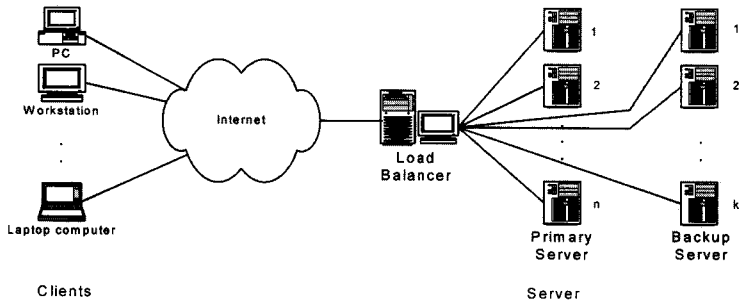
소프트웨어는 동작 시점에는 오류없이 거의 완벽한 성능을 갖더라도 시간이 지남에 따라, 즉 소프트웨어의 노화(Software Aging)로 인해 시스템 성능 저하 및 일시적 결함의 발생 가능성이 커지게 되며, 결국에는 소프트웨어의 성능이 점차로 저하되어, 이 같은 현상이 지속될 경우 시스템 작동이 멈추게 된다[9]. 이런 문제를 해결하기 위하여 소프트웨어 재활(Software Rejuvenation) 기법을 사용한다. 소프트웨어 재활 기법은 시스템의 결함 발생 이후에 수동적으로 대처하기 보다는 결함이 발생하기 전에 이를 미연에 방지하는 능동적 차원의 결함 허용 방법이다[6,13].

본 연구에서는 n대의 주 서버와 k대의 여분 서버를 가진 클러스터 시스템에 재활 기법을 적용하여, 시스템에서 수행되는 소프트웨어의 재활주기, 재활소요시간, 서버의 고장율, 수리율, 주 서버의 수, 여분 서버의 수, 서버의 가동시간 등의 시스템 운영파라미터에 기초하여 소프트웨어 재활 정책을 수학적 방법으로 계산하였다.

본 논문의 2장에서는 재활기법에 의한 (n,k) 클러스터 시스템 모델을 정의하고, 3장에서는 제안된 모델의 수학적 해석 방법의 정확성을 검증하기 위하여 다양한 운영 조건에 대한 실험을 수행하였다. 마지막으로 결론에서 제안된 소프트웨어 재활 기법의 활용 방안 및 향후 연구에 관하여 논하였다.

2. (n,k) 클러스터 시스템 모델

본 연구에서 분석하고자 하는 (n,k) 클러스터 시스템은 [그림 1]과 같다. (n,k) 클러스터 시스템은 n대의 주 서버(Primary Server)와 k대의 여분서버(Backup Server)로 구성되어 있으며, 임의의 주 서버에 고장이 날 경우 k대의 여분서버중 한 대가 주 서버의 역할을 대신하는 시스템이다. 로드 밸런서(Load



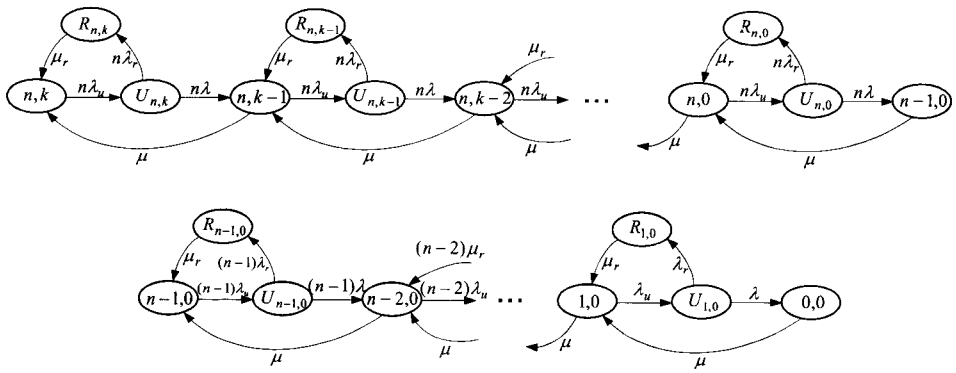
[그림 1] (n,k) 클러스터 시스템 구성도

[Fig. 1] The Configuration diagram of (n,k) cluster system

Balancer)는 클러스터를 구성하는 컴퓨터 중 하나가 그 역할을 대신한다.

현재까지 컴퓨터 시스템의 가용도에 관한 연구들은 시스템의 성능(대기시간, 응답시간)을 고려하지 않은 가용도 계산 모델이 주를 이루고 있다[9,10,11]. 본 논문에서는 n대의 주 서버와 k대의 여분 서버를 두고 재활과 가용도를 고려한 모델로, (n,k) 클러스터 시스템에 사용될 가정을 다음과 같이 정의한다.

- n 대의 주 서버가 클러스터로 연결되어 있으며, 각 서버는 서로 독립적으로 작업을 수행한다.
- k 대의 여분서버가 존재하며($n \geq k$), 임의의 주 서버에 고장이 날 경우 이의 역할을 Active/Standby 방식으로 대체한다.
- 백업서버는 고장나지 않으며, 주 서버가 고장 날 경우 즉각 여분 서버로 교체되며, 이 시간은 극히 짧은 시간으로 무시할 수 있다.
- 주 서버와 여분서버의 고장률(λ), 수리률(μ)은 상수이다. 즉 고장 및 수리 시간 간격은 지수분포를 따른다.
- 여분서버는 주 서버 고장 시에만 사용되며, 고장 난 서버는 수리된다.
- 재활에 들어갈 경우, k대의 여분 서버를 제외한 n대의 주 서버가 작업 대상이며, 재활 작업시간은 여분 서버 수에 무관하다.
- 장시간 가동으로 인한 서버의 불안정율(λ_u)은 모든 가동 상태에서 동일하다.
- 서버의 가동을 주기적으로 멈추는 재활율(λ_r)은 모든 가동상태에서 동일하다.



[그림 2] (n,k) 클러스터 시스템의 고장-재활-수리 상태 전이도

[Fig. 2] The Fault-Rejuvenation-Repair state transition diagram of (n,k) cluster system

본 논문에서 제시한 (n, k) 클러스터 시스템의 상태 모델을 [그림 2]에 나타내었다. 정상상태에서 가동중인 서버는 $(n,k),(n,k-1),\cdots,(n,0),(0,0)$ 등의 가동중인 (주 서버, 여분 서버)의 수를 상태 변수로 가지고 있으며, k개의 여분서버가 모두 사용될 때까지는 시스템의 가동서버는 n개로 유지된다. (0,0)인 상태는 주 서버와 여분서버의 가동이 모두 중지된 상태로 고장상태를 의미한다.

정상상태에서 가동중인 서버가 장시간 가동으로 인해 성능이 저하된 상태(Ustable State)는 $U_{n,k}, U_{n,k-1},\cdots,U_{n,0},\cdots,U_{1,0},U_{0,0}$로 나타내었다. 정상상태에서 불안정상태로 변화율을 λ_u로 표시하며, 이는 소프트웨어의 장기간 가동으로 인한 시스템의 불안정성을 반영한다. 불안정상태(Unstable State)에서 λ_r의 변화율로 재활작업에 들어가거나, $n\lambda$(여분의 서버가 존재할 경우 n대의 주 서버가 가동함)나 $i\lambda$(여분서버가 없을 경우 i대의 주 서버가 가동함)의 변화율로 고장이 발생하게 된다. 재활상태(Rejuvenation State)는 $R_{n,k}, R_{n,k-1},\cdots,R_{n,0},\cdots,R_{1,0},R_{0,0}$으로 표시하며, 시스템의 가동을 고의로 중지시켜 재 부팅하는 상태(State)를 나타낸다. 여분서버가 없을 경우(k=0) i대의 클러스터 중 임의의 서버에서 결함이 발생할 경우 이를 감지하여 클러스터로부터 결함 서버를 제거하는 결함제거 상태로 들어가게 된다. [그림 1]의 평형상태(Steady State)에서의 균형식(Balance Equation)은 다음과 같다.

$$n\lambda_u P_{n,k}=\mu_r P_{R_{n,k}}+\mu P_{n,k-1} \tag{1}$$

$$(\mu+n\lambda_u)P_{n,j}=\mu_r P_{R_{n,j}}+n\lambda P_{U_{n,j+1}}+\mu P_{n,j-1} \quad j=1,\cdots,k-1 \tag{2}$$

$$(\mu+n\lambda_u)P_{n,0}=\mu_r P_{R_{n,0}}+n\lambda P_{U_{n,1}}+\mu P_{n-1,0} \tag{3}$$

$$(\mu+i\lambda_u)P_{i,0}=\mu_r P_{R_{i,0}}+(i+1)\lambda P_{U_{i+1,0}}+\mu P_{i-1,0} \quad i=1,\cdots,n-1 \tag{4}$$

$$\mu P_{0,0}=\lambda P_{U_{1,0}} \tag{5}$$

$$n(\lambda+\lambda_r)P_{U_{n,j}}=n\lambda_u P_{n,j} \qquad j=0,\cdots,k \tag{6}$$

$$i(\lambda+\lambda_r)P_{U_{i,0}}=i\lambda_u P_{i,0} \qquad i=1,\cdots,n-1 \tag{7}$$

$$\mu_r P_{R_{n,j}}=n\lambda_r P_{U_{n,j}} \qquad j=0,\cdots,k \tag{8}$$

$$\mu_r P_{R_{i,0}}=i\lambda_r P_{U_{i,0}} \qquad i=1,\cdots,n-1 \tag{9}$$

$$\sum_{j=0}^{k}P_{n,j}+\sum_{i=0}^{n-1}P_{i,0}+\sum_{j=0}^{k}P_{U_{n,j}}+\sum_{i=1}^{n-1}P_{U_{i,0}}+\sum_{j=0}^{k}P_{R_{n,j}}+\sum_{i=1}^{n-1}P_{R_{i,0}}=1 \tag{10}$$

위의 Balance Equation과 각 State에서 머물 확률의 총합이 1이 되는 식을 결합한 연립방정식을 풀면, 시스템이 평형일 때 각 상태에서 머물 확률을 얻을 수 있다.

$$P_{n,k}=\left[\left(1+\frac{\lambda_u}{\lambda+\lambda_r}+\frac{\lambda_r}{\mu_r}\frac{n\lambda_u}{\lambda+\lambda_r}\right)\sum_{j=0}^{k}\left(\frac{\lambda}{\mu}\right)^{k-j}\left(\frac{n\lambda_u}{\lambda+\lambda_r}\right)^{k-j}+\left(\frac{\lambda}{\mu}\right)^{n+k}\left(\frac{\lambda_u}{\lambda+\lambda_r}\right)^{n+k}n^k n!+\left(\left(1+\frac{\lambda_u}{\lambda+\lambda_r}\right)\sum_{i=1}^{n-1}\left(\frac{\lambda}{\mu}\right)^{n-i}\left(\frac{\lambda_u}{\lambda+\lambda_r}\right)^{n-i}\frac{n!}{i!}+\left(\frac{\lambda_r}{\mu_r}\frac{\lambda_u}{\lambda+\lambda_r}\right)\sum_{i=1}^{n-1}i\left(\frac{\lambda}{\mu}\right)^{n-i}\left(\frac{\lambda_u}{\lambda+\lambda_r}\right)^{n-i}\frac{n!}{i!}\right)\left(\frac{\lambda}{\mu}\frac{n\lambda_u}{\lambda+\lambda_r}\right)^{k}\right]^{-1} \quad n>k, n=1,\cdots \tag{11}$$

$$P_{n,0}=\left(\frac{\lambda}{\mu}\right)^{k}\left(\frac{n\lambda_u}{\lambda+\lambda_r}\right)^{k}P_{n,k} \tag{12}$$

$$P_{n,j}=\left(\frac{\lambda}{\mu}\frac{n\lambda_u}{\lambda+\lambda_r}\right)^{k-j}P_{n,k} \quad j=0,\cdots,k \tag{13}$$

$$P_{i,0}=\left(\frac{\lambda}{\mu}\frac{\lambda_u}{\lambda+\lambda_r}\right)^{n-i}\frac{n!}{i!}\left(\frac{\lambda}{\mu}\frac{n\lambda_u}{\lambda+\lambda_r}\right)^{k}P_{n,k} \quad i=0,\cdots,n-1 \tag{14}$$

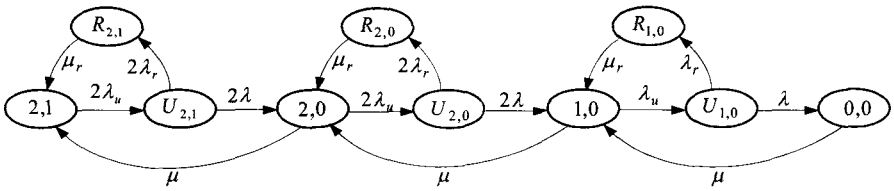
$$P_{R_{n,j}}=\frac{n\lambda_r}{\mu_r}\frac{\lambda_u}{\lambda+\lambda_r}P_{n,j} \qquad j=0,\cdots,k \tag{15}$$

$$P_{R_{i,0}}=\frac{i\lambda_r}{\mu_r}\frac{\lambda_u}{\lambda+\lambda_r}P_{i,0} \qquad i=1,\cdots,n-1 \tag{16}$$

$$P_{U_{n,j}}=\frac{\lambda_u}{\lambda+\lambda_r}P_{n,j} \qquad j=0,\cdots,k \tag{17}$$

$$P_{U_{i,0}}=\frac{\lambda_u}{\lambda+\lambda_r}P_{i,0} \qquad i=1,\cdots,n-1 \tag{18}$$

각 시스템의 운영파라미터를 이용하여 $P_{n,k}$와 $P_{n,0}$를 계산하여 그 외의 정상상태 $(P_{n,j},P_{i,0})$와



[그림 3] (2,1) 클러스터 시스템의 고장-재활-수리 상태 전이도

[Fig. 3] The Fault-Rejuvenation-Repair state transition diagram of (2,1) cluster system

재활상태 ($P_{R_{n,j}}, P_{R_{i,0}}$) 그리고 불안정상태 ($P_{U_{n,j}}$, $P_{U_{i,0}}$)에서의 확률을 구할 수 있다.

[그림 3]은 주 서버가 2대이고 여분서버가 1대인 (3,1) 클러스터 시스템의 상태 모델의 예이다.

3. 실험 및 성능평가

① 가용도 : 클러스터 시스템을 구성하는 서버의 가동 여부만을 고려하여 가용도는 모든 서버가 고장난 상태 ($P_{0,0}$)이거나 한 대의 서버만 가동중인 상태에서 재활작업 ($P_{R_{1,0}}$)을 수행하는 상태를 제외함으로써 계산될 수 있다.

$A = 1 - (P_{0,0} + P_{R_{1,0}})$

$UA = P_{0,0} + P_{R_{1,0}}$

② Downtime : 재활 작업으로 인해 서비스를 받을 수 없는 Downtime은 가동시간(T)에 대한 함수로 다음과 같다.

D(T) = $UA * T$

③ 손실비용 : 서버의 가동정지로 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 저렴하다.

$Cost(T) = (P_{0,0} * C_F + P_{R_{1,0}} * C_R) * T$

④ 시스템 운영 파라미터 : 본 논문에서 사용한 다양한 시스템 운영 파라미터는 그림 4와 같다. 2대의 주 서버와 1대의 여분 서버의 가동기간(T) 1년으로 하며, 서버의 고장 (λ)은 6개

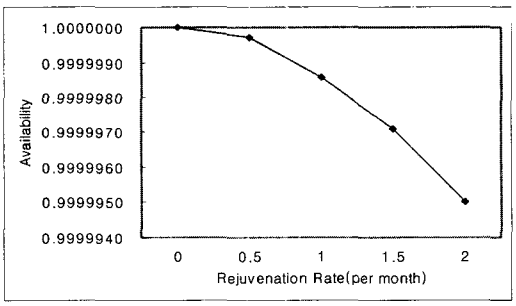
주 서버의 수(n)	2	number of primary servers
여분 서버의 수(k)	1	number of backup servers
가동기간(T)	12*30*24*3600	1 year
고장률 (λ)	1/(6*30*24*3600)	2 time / year
수리율 (μ)	1/(60*120)	1 time / 2 hours
재활율 (λ_r)	1/(30*24*3600)	1 time / month
재활작업시간 (μ_r)	2/(60*60)	2 times / hour
불안정률 (λ_u)	1/(3*24*3600)	every 3 days
불시정지로 인한 다운 비용 (C_F)	1000	downtime cost
재활로 인한 다운 비용 (C_R)	1	rejuvenation cost

[그림 4] 시스템 운영 파라미터

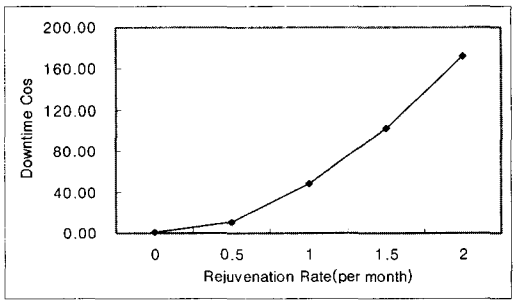
[Fig. 4] System operation parameters

월에 1회, 고장수리에 (μ)에 2시간이 소요되며, 한 달에 한 번씩 재활 작업 (λ_r)을 수행하며 30분이 소요된다. 시스템 불시정지로 인해 발생하는 비용 (C_f)은 재활 작업으로 발생하는 비용 (C_r)의 1000배가 된다. 또한 이와 같은 파라미터 값들은 여러 가지 실험에서 다양하게 변화 시켜가며 실험하였다.

[그림 5]는 재활작업을 수행하는 기간에 따른 변화로, 재활 작업을 빈번하게 수행하게 되면 가용도가 감소하며, 손실비용은 증가하는 추세를 보이고 있다. [그림 6]은 시스템의 고장이 얼마나 자주 일어나느냐에 따른 결과로, 고장이 적을수록 가용도는 증가하는 추세를 보이며, 이에 따른 손실비용도 감소하고 있다. 특히 3개월을 기준으로 눈에 띄게 변화하는 추세를 보이고 있다.



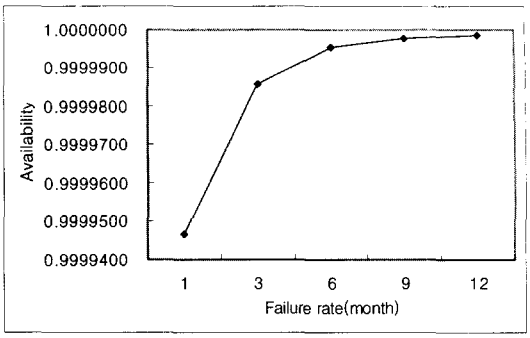
(a) 가용도(Availability)



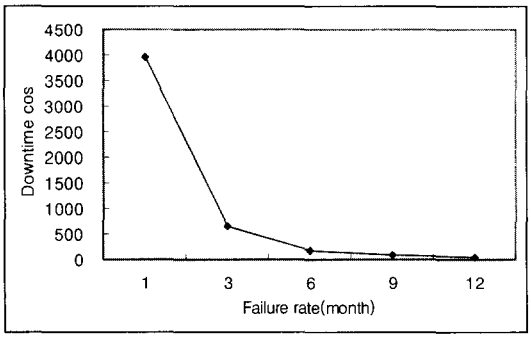
(b) 손실비용(Downtime cost)

[그림 5] 재활률 (λ_r)에 따른 가용도 및 손실비용

[Fig. 5] Availability and Downtime cost according to rejuvenation rate (λ_r)



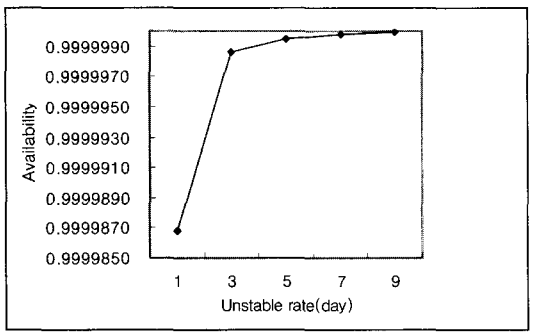
(a) 가용도(Availability)



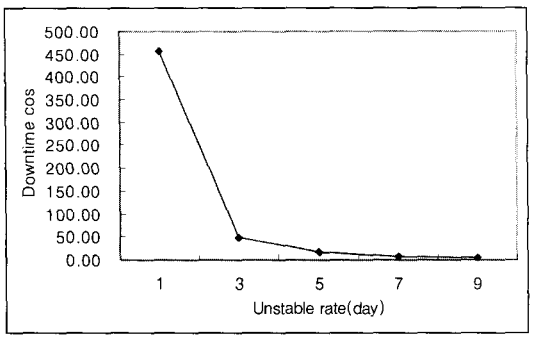
(b) 손실비용(Downtime cost)

[그림 6] 고장률 (λ)에 따른 가용도 및 손실비용

[Fig. 6] Availability and Downtime cost according to failure rate (λ)



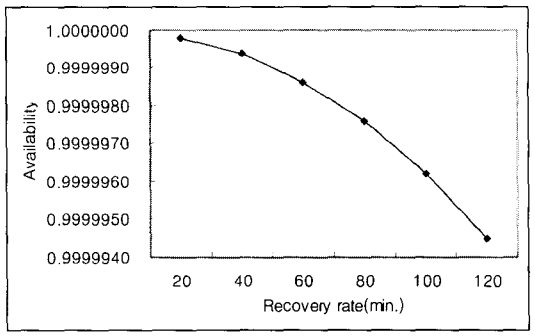
(a) 가용도(Availability)



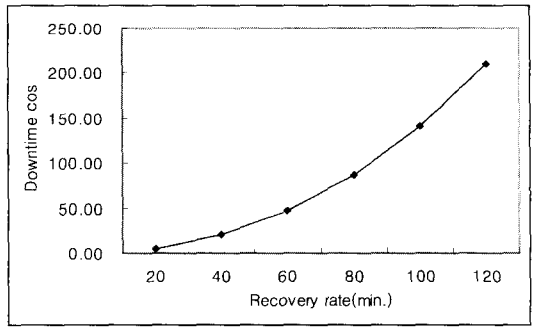
(b) 손실비용(Downtime cost)

[그림 7] 불안정률 (λ_u)에 따른 가용도 및 손실비용

[Fig. 7] Availability and Downtime cost according to unstable rate (λ_u)



(a) 가용도(Availability)



(b) 손실비용(Downtime cost)

[그림 8] 수리율 (μ)에 따른 가용도 및 손실비용

[Fig. 8] Availability and Downtime cost according to repair rate (μ)

[그림 7]은 시스템의 불안정 정도에 따른 변화로, 시스템이 빨리 불안정 상태로 변하게 되면 가용도는 감소하고 안정적인 시스템일수록 가용도는 높아지는 추세를 보이고 있으며, 손실 비용도 시스템이 안정적일수록 낮아지는 추세를 보이고 있다. 특히 3일 기준으로 급격히 가용도는 증가하고, 손실비용은 감소하고 있다. [그림 8]은 시스템의 수리율에 따른 변화로 수리하는데 걸리는 시간이 길면 길수록 가용도는 감소하는 추세를 보이며, 손실 비용은 증가하는 추세를 보이고 있다.

4. 결론

본 연구에서는 웹 서버를 클러스터로 사용하는 시스템에서 가동되는 서버의 수, 여분서버의 수, 소프트웨어의 재활주기, 재활소요시간, 서버의 고장률, 서버의 수리율, 서버의 불안정률 등의 시스템 운영 파라미터에 기초하여, 소프트웨어의 재활정책에 대한 평가를 위해 평형 상태에서의 확률, 정지시간, 가용도, 손실비용 등을 계산하였다. 수학적 분석을 통해 다양한 시스템 운영 상태에 대한 실험을 통해 검증하였으며, 소프트웨어의 재활 정책에 의한 예방적 결함허용 기법이 시스템의 안정성에 중요한 요소임을 확인하였고, 여분 서버를 두어 서버를 운영하는 것이 가용도를 높이는데 중요한 요소임을 파악하였다. 또한 서버의 고장률 및 불안정률이 소프트웨어 재활 정책 결정에 중요한 요소임을 파악하였다. 따라서 무정지로 운영되어야 하는 웹서버의 경우 소프트웨어의 빠른 노화로 가용도가 급격히 감소하게 되는데, 이에 대한 예방적 차원에서 가용도를 높이며, 유지보수 비용을 상당히 줄일 수 있는 방법이라 본다.

추후에는 시스템이 고장이 발생할 경우 여분 서버로 대치하는 데 걸리는 작업 전이 시간을 고려할 필요성이 있고, 재활 상태를 지수분포가 아닌 다른 분포를 사용하여 좀더 일반화 할 필요가 있으며, 시스템의 성능을 포함하는 모델을 통한 가용도 분석과 소프트웨어 재활 정책을 포함하는 가용도 모델의 정립과 분석이 필요하다고 본다.

※참고 문헌

[1] 권세오, 김상식, 김동승, "리눅스 클러스터형 웹 서버 설계," 정보과학회지, 제18권, 제3호, pp. 48-56, 2000. 3.

[2] 박기진, 김성수, 김재훈, "소프트웨어 재활 기법을 적용한 다중계 시스템의 가용도 분석," 한국정보과학회논문지(시스템 및 이론), 제27권, 제8호, pp. 730-740, 2000. 8.

[3] 오수철, 정상화, "클러스터 시스템 기술 동향," 정보과학회지, 제18권, 제3호, pp.4-10, 2000. 3.

[4] 유찬수, "리눅스 클러스터링," 정보과학회지, 제18권, 제2호, pp. 33-39, 2000. 2.

[5] B. Krishnamurthy and C.E. Wills, "Analyzing Factors that Influence End-To-End Web Performance," Computer Networks, Vol. 33, pp. 17-32, 2000.

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

[7] D. Anderson, T. Yang and O.H. Ibarra, "Toward a Scalable Distributed WWW Server on Workstation Clusters," Journal of Parallel and Distributed Computing, Vol. 42, pp. 91-100, 1997.

[8] F. Wang, K. Ramamritham and J.A. Stankovic, "Determining Redundancy Levels for Fault Tolerant Real-Time Systems," IEEE Transactions on Computers, Vol. 44, No. 2, pp. 292-301, Feb. 1995.

[9] K. Vaidyanathan, R. E. Harper, S. W. Hunter and K. S. Trivedi, "Analysis and Implemantation of Software Rejuvenation in Cluster Systems," ACM SIGMETRICS 2001/Performance 2001, June 2001

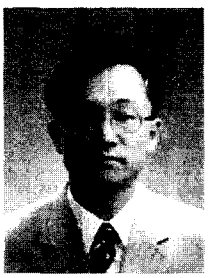
[10] M. Nabe, M. Murata and H. Miyahara, "Analysis and Modeling of World Wide Web Traffic for Capacity Dimensioning of Internet Access Lines," Performance Evaluation, Vol. 34, pp. 249-271, 1998.

[11] M.S. Squillante, D.D. Yao and L. Zhang, "Web Traffic Modeling and Web Server Performance Analysis," Proc. of the 38th Conference on Decision & Control, Phoenix, Arizona, pp. 4432-4439, Dec. 1999.

[12] R. Buyya, *High Performance Cluster Computing Volume 1: Architectures and Systems*. p. 849, Prentice-Hall, 1999.

[13] Y. Huang, C. Kintala, N. Kolettis and N. Fulton, "Software Rejuvenaion: Analysis, Module and Applications," IEEE Intl. Symposium on Fault Tolelant Computing, FTCS 25, pp. 381-390, June 27-30, 1995.

강 창 훈



1986 충남대학교 계산통계학과 졸업(이학사)
1988 충남대학교 대학원 계산통계학과 졸업(이학석사)
1999 아주대학교 대학원 컴퓨터공학과 수료
1994～현재 극동정보대학 멀티미디어과 조교수
관심분야 : 결함허용, 성능분석, 클러스터컴퓨팅, 멀티미디어시스템