가용성이 높은 Cold Standby 클러스터 시스템의 성능 분석

(Performance Analysis of Highly Available Cold Standby Cluster Systems)

박 기 진 † 김 성 수 ††

(Kiejin Park) (Sungsoo Kim)

요 약 고가용도 클러스터 시스템에서 가동되는 인터넷 기반 소프트웨어의 복잡도가 증가됨에 따라 소프트웨어의 설계, 구현, 또는 그 밖의 여러 가지 원인과 관련된 결함으로 인하여 시스템 서비스의 오동작 또는 수행 중단으로 이어지는 사례가 늘어나고 있다. 특히 대량 트랜잭션을 처리하는 인터넷 기반 컴퓨팅 소프트웨어는 빈번한 통신 두절과 데이타 유실로 인하여, 이들이 탑재된 클러스터 시스템의 결함 발생이 더욱 심각할 가능성이 높다. 본 연구는 소프트웨어 재활 결함 허용 기법을 활용하여, 별도의 추가되는 하드웨어 없이도 가용도를 개선할 수 있다는 "소프트웨어 재활 기법을 적용한 다중계 시스템 가용도 분석" 에 관한 논문에서 언급된 문제점들에 대한 해결 방안을 제시하였으며, 구체적으로는 1) 주서버의 고장 발생시 여분서버로의 작업전이(switchover) 상태를 클러스터 시스템 모델링에 포함시켰으며, 2) 작업전이 상태와 재활(rejuvenation) 상태에서 머무는 시간을 지수분포 대신에 k-stage Erlangian 분포를 사용하여 확정시간(deterministic time)을 표현할 수 있도록 하였다. 즉 본 논문에서는 고가용도 cold standby 클러스터 시스템의 운영 상태에 대한 상태전이도(state transition diagram)에서, 임의의 상태에서 머무는 시간분포가 memoryless 성질을 만족하지 않아도 되는 semi-Markov 프로세스 문제를 해결하였다.

Abstract Due to the complexity of Internet based computing softwares in which have some undetected design or implementation faults, highly available cluster systems that run the software become unreliable or stop service frequently. Especially, because of the loss of communications or data, Internet based computing software that deals with massive transaction processing jobs may easily be unavailable. In this paper, we solve the specific problems left as the further research topics in our previous publication that demonstrates the possibility of availability improvement for multiplex systems using software rejuvenation method. Those are 1) we include the switchover state in our model to represent the role of the primary and backup server, 2) instead of using memoryless property to describing sojourn time distribution for a rejuvenation state and a switchover state, we use k-stage Erlangian distribution and then it becomes possible to give deterministic time for the sojourn time of rejuvenation or switchover. In other words, our state transition diagram for a highly available cold standby cluster systems consists of two kinds of states that are exponentially distributed sojourn time and deterministic sojourn time. This kind of problem is classified as the semi-Markov process.

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

† 학생회원 : 아주대학교 컴퓨터공학과
kiejin@yahoo.co.kr
†† 종신회원 : 아주대학교 정보통신전문대학원 교수
sskim@madang.ajou.ac.kr

논문접수 : 2000년 8월 1일
심사완료 : 2000년 11월 23일

1. 서 론

최근 인터넷과 관련된 여러 가지 종류의 응용 서비스가 급격히 증가하고 있는 추세이며, 인터넷을 통해서 상품이나 서비스를 제공하는 기업은 더욱 늘어날 전망이다[1,2]. 인터넷을 통해 상품이나 서비스를 제공하는 새로운 비즈니스 활동이 이루어지는 과정을 살펴보면, 크게 소비자와 서비스 제공자 사이의 거래가 이루어지는

부분과 서비스 제공자와 다른 서비스 제공자(인증, 지불 기관) 사이의 연결을 시켜주는 두 부분으로 나누어지며, 인터넷 기반 비즈니스 서비스를 제공하려는 기업 입장에서는 이러한 새로운 비즈니스 모델 활동을 지원하기 위해서는 반드시 대규모 인터넷 서버를 구축하고 유지해야 한다. 향후 거의 모든 경제활동이 인터넷 서버를 통해서 이루어진다는 점을 고려하면 서비스 시스템을 구축할 때 가장 주안점을 두어야 할 것이 바로 시스템의 가동 능력 즉 가용도이며, 서버는 하루 24 시간동안 계속해서 가동되어야 하므로 시스템을 구축할 때 서버의 가용도를 높이기 위한 비용 효율적인 방법을 적용해야만 한다[3,4].

또 한편으로는 전자상거래 서버, 웹서버, CTI(computer telephony integration) 서버 등 여러 분야에서 고성능 서버에 대한 수요의 증가와 더불어 고성능 서버의 비용 문제를 해결하는 방안으로 클러스터 시스템에 많은 관심이 집중되고 있다[5,6]. 리눅스 고가용도 클러스터 시스템 프로젝트는 PC 또는 워크스테이션을 네트워크로 연결하여 고성능 서버의 성능을 발휘하는 시스템을 제작하는 것으로, 이와 같은 시스템의 성공 여부는 네트워크 성능, 지원 소프트웨어의 수준은 물론, 다수의 컴퓨터를 동시에 가동할 때 발생하는 시스템의 가용도 저하 문제의 해결에 있다[7,8,9].

가용도를 높이기 위한 방법으로 현재 가동중인 주서버(primary server)를 대체할 수 있는 여분서버(backup server)를 두는 방법이 주로 사용되며, 주서버와 여분서버간의 작업전이 방식에 따라 HSS(hot standby system)와 CSS(cold standby system)로 구분하고 있다[10]. 위의 두 가지 방법 중에서 HSS는 주서버를 대체하는 시간이 매우 짧지만 여러 대의 서버를 동시에 가동해야 하므로, CSS에 비해서 비용이 많이 들며, 또한 장시간 가동으로 인해 서버의 고장 발생률이 커지게 된다. 반면에 CSS의 경우는 HSS에 비해서 주서버의 역할을 대체하는데 소요되는 작업전이(switchover) 시간은 길지만 그만큼 비용이 적게 들고 서버의 고장 발생률이 상대적으로 적어진다. 따라서 가용도가 엄격한 시스템(교환기, CTI 서버, 멀티미디어 서버)에서는 HSS를 사용해야 하지만 어느 정도까지 허용하는 시스템(전자상거래서버, 웹서버)에서는 CSS를 사용하는 것이 바람직하다.

클러스터 시스템은 사용 목적에 따라 크게 고가용성 클러스터와 부하 분산 클러스터로 구분할 수 있다[11]. 전자는 서버들의 일부에서 결함이 발생했을 때 이를 허용하는 것이 그 목적으로, 고장난 서버를 클러스터에서

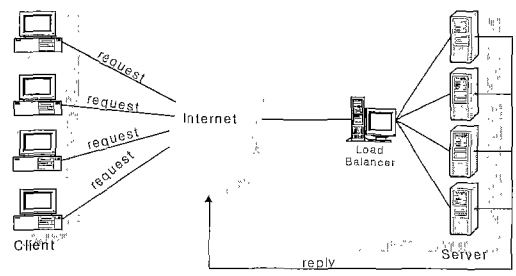

그림 1 부하분산 클러스터 시스템의 서버 연결구조

동적으로 제거하고 해당 서버가 수행하던 작업을 다른 컴퓨터가 대신 수행하며, 고장 수리를 완료한 이후에는 다시 클러스터 시스템으로 합류할 수도 있다. 후자는 웹, 멀티미디어, 및 FTP 등의 서비스 요청을 받으면, 시스템 내에서 부하가 가장 작은 서버에게 서비스를 제공하게 함으로 대규모 서비스를 가능하게 한다. 예를 들면, 소규모 인터넷 서비스 시스템의 가용도를 확보하기 위해서 2대의 서버로 구성된 클러스터 시스템을 구축할 경우, 거의 무정지로 서비스를 제공할 수 있다. 그림 1은 부하분산 클러스터형 시스템의 연결구조 모습이다. 실제 데이타를 저장하고 있는 서버들의 작업부하는 부하 분배작업을 하는 가상 서버(load balancer)에 의해 클라이언트와 연결되는 구조를 가지고 있다. 이러한 구조의 시스템에서는 서버의 가동 대수가 증가함으로 인해 발생하는 가용도 저하 문제와 가상 서버의 단일점 고장(single point of failures)으로 인한 시스템 다운의 위험이 발생하게 된다[12].

클러스터 시스템에서 가동되는 인터넷 기반 소프트웨어의 복잡도가 증가됨에 따라 소프트웨어의 설계, 구현, 또는 그 밖의 여러 가지 원인과 관련된 결함으로 인하여 시스템 서비스의 오동작 또는 수행 중단으로 이어지는 사례가 늘어나고 있다. 특히 대량 트랜잭션을 처리하는 인터넷 기반 컴퓨팅 소프트웨어는 빈번한 통신 두절과 데이타 유실로 인하여, 이들이 탑재된 클러스터 시스템의 결함 발생이 더욱 심각할 가능성이 높다[13,14,15,16]. 그러나, 지금까지의 많은 연구들이 대부분 하드웨어의 결함을 줄이기 위한 것이고 소프트웨어적 원인으로 인해 발생되는 결함을 줄이기 위한 연구는 상대적으로 미진한 실정이다[17,18,19]. 본 논문에서는 위와 같은 문제점들에 대한 해결 방안의 일환으로, 다수의 서버로 구성된 인터넷 기반 고가용도 cold standby 클러스터 시스템의 결함을 사전에 예방할 수 있고, 별도의 하드웨어 자원을 요구하지 않으며, 결함으로 인해 발

생하는 손실 비용을 최소화 할 수 있는 소프트웨어 재활 결함 예방 기법에 관하여 연구하였다. 하드웨어 기술의 발전으로 인해 하드웨어의 결함 발생률은 상수 값이거나 점차 작아지는 경향(즉 시스템의 하드웨어 신뢰도만 증가)이 있는 반면에 응용 프로그램의 복잡성 및 크기는 이전에는 상상할 수 없을 정도로 방대해져가고 있기 때문에, 향후에는 시스템을 구성하고 있는 하드웨어는 물론 응용 소프트웨어의 가용성을 높이는 연구가 활발할 것으로 예상된다. 하드웨어 및 소프트웨어로 구성된 전체 시스템의 의존도(dependability)는 어느 한쪽의 가용도 혹은 신뢰도를 보장한다고 해서 완벽할 수는 없기 때문에, 하드웨어에 대한 결함허용과 동시에 이 하드웨어 상에서 가동되는 대형 응용 프로그램의 수행도 보장해야만 한다[20].

2. 관련 연구

클러스터 시스템의 결함으로 인해 발생하는 손실 비용을 최소화하기 위한 소프트웨어 재활 기법은 시스템의 결함 발생 이후에 수동적으로 대처하기보다는 결함이 발생하기 전에 이를 미연에 방지하는 능동적 차원의 결함 허용 방법이다. 특히 인터넷 기반 소프트웨어는 통신 단절, 데이타 유실 등으로 인한 노화(memory leak, accumulated round off error, buffer overflow 등) 진행이 일반 소프트웨어보다 상당히 빠르게 진행되기 때문에 소프트웨어 재활에 의한 결함 예방 방법은 대규모 인터넷 기반 고가용도 클러스터 시스템에 사용될 가능성이 높다고 볼 수 있다. 본 연구에서는 이미 발표된 본 저자들의 연구 논문[21]에서 언급된 문제점에 대한 해결 방안을 제시하였으며, 구체적으로는 1) 주서버의 고장 발생 시 여분서버로의 작업전이 상태를 클러스터 시스템 모델링에 포함시켰으며, 2) 재활(rejuvenation) 상태와 작업전이 상태에서 머무는 시간을 지수분포 대신에 k-stage Erlangian 분포를 사용하여 확정시간(deterministic time)을 표현할 수 있도록 하였다. 본 논문의 2장에서는 관련 연구 결과와 문제를 정의하였으며, 3장에서는 소프트웨어 재활 기법을 적용한 고가용도 cold standby 클러스터 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. cold standby 클러스터 시스템 모델

cold standby 클러스터 시스템 모델링에 사용된 기본적인 가정들은 다음과 같다.

- n개의 서버로 구성된 고가용도 cold standby 클러스터 시스템에서 각 서버의 고장률(λ)은 동일하다.
- 고장난 서버를 수리하는 수리률(μ)은 동일하다.
- 서버의 가동을 주기적으로 멈추는 재활률(λ_r)은 모든 가동 상태에서 동일하다.
- 장시간 가동으로 인한 서버의 불안정률(λ_u)은 모든 가동 상태에서 동일하다.
- 재활 작업 시간은 평균값 $1/\mu_r$ 인 k-stage Erlangian 분포를 따른다.
- 주서버에서 여분서버로의 작업전이 시간은 평균값 $1/\lambda_s$ 인 k-stage Erlangian 분포를 따른다.
- 재활 상태와 작업전이 상태를 제외한 모든 상태에서 머무는 시간은 지수분포를 따른다.
- 재활에 들어갈 경우, 현재 가동하고 있는 주서버가 재활 작업 대상이며, 가동되고 있지 않는 서버는 고려 대상에서 제외된다.

cold standby 가동 방식에 따른 소프트웨어 재활을 고려한 고가용도 cold standby 클러스터 시스템의 상태 모델을 그림 2에 나타내었다. 정상 상태(normal state)에서 가동되고 있는 서버는 n, n-1, ..., 1, 0 등의 가동중인 서버의 수를 상태 변수로 가지고 있으며, 장시간 가동으로 인해 성능이 저하된 불안정 상태(unstable state)의 서버는 u_n, u_{n-1}, ..., u_2, u_1로 표시된다. 정상 상태에서 불안정 상태로의 변화율은 λ_u로 표시되며, 이는 소프트웨어의 장기간 가동으로 인한 시스템의 불안정률을 반영한다. 불안정 상태에서는 λ_r의 변화율로 재활 상태에 들어갈 경우, 소프트웨어 노화 현상으로 인해 발

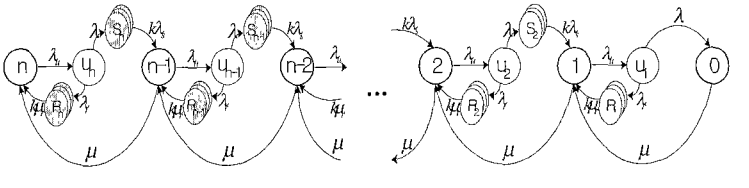

그림 2 cold standby 클러스터 시스템 상태 전이도(state transition diagram)

생되는 일시적 결함은 대부분 제거될 수 있다. CSS의 경우, 특정 시점에서 가동되고 있는 서버는 오직 한 대이기 때문에 불안정 상태(u_i)에서 고장 발생률(λ)이 일정하게 유지되도록 모델링 된다. 주서버에 고장이 발생할 경우 여분서버에서 작업을 대체 수행하는 작업전이 상태를 회색의 겹쳐진 원으로 표시된 S_n, S_{n-1}, ..., S_3, S_2 로 나타냈다. 또한 회색의 겹쳐진 원모양의 R_n, R_{n-1}, ..., R_2, R_1은 재활 상태를 표시하며, 시스템의 가동을 고의로 중지시키는 현상을 나타낸다. 그림 2의 모델링은 재활 상태와 작업전이 상태에서 머무는 시간(sojourn time)의 분포가 memoryless 성질을 만족하지 않기 때문에, semi-Markov 프로세스 문제로 분류되어, 정확한 해를 구하는 간단한 방법이 존재하지 않는다[22].

일반적으로 시스템을 재활하여, 깨끗한 상태로 만드는데 소요되는 시간(rebooting time 혹은 purging time)은 일정한 상수 값을 가지며, 주서버에서 여분서버로의 작업전이 시간 또한 일정 시간 이내에 이루어져야 하는 조건을 가지고 있다. 이와 같이 현실적인 고가용성 클러스터 시스템의 운영 상태를 반영하는 모델의 수학적 해를 구하기 위해서, 그림 2에서 회색으로 칠해진 재활 상태와 작업전이 상태에서 머무는 시간을 임의의 수 k개로 나눈 분할 상태 전이도로 모델링 하였다(그림 3 참조).

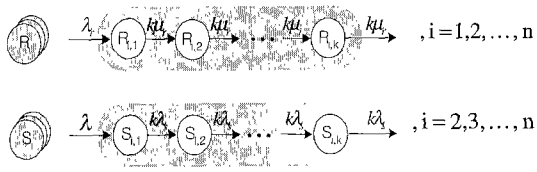

그림 3 sojourn time의 분할 상태 전이도

임의의 재활 상태 R_i 는 k 개의 분할 상태를 가지는 부분재활 상태(sub-rejuvenation state: $R_{i,j}$)들로 표현되며, 각 부분재활 상태에서의 재활 소요 시간은 평균 $1/k\mu_r$ 인 지수분포를 따른다. 부분재활 상태의 분포함수 및 평균재활 시간은 다음과 같다.

$$h(y)=k\mu_r e^{-k\mu_r y}, E(\bar{y})=\frac{1}{k\mu_r}, y\geq 0$$

k개로 분할된 재활 상태 R_i 에서 머무는 평균 시간은 각 stage의 부분재활 상태에 머무는 평균 시간으로부터 계산된다.

$$E(\bar{x})=k\left(\frac{1}{k\mu_r}\right)=\frac{1}{\mu_r}$$

그림 3의 분할 상태 전이도는 k 개의 지수분포의 합, 즉 k-stage Erlangian 분포이며 다음과 같다[23].

$$b(x)=\frac{k\mu_r(k\mu_r x)^{k-1}e^{-k\mu_r x}}{(k-1)!}, x\geq 0$$

지수분포의 합으로 이루어진 k-stage Erlangian 분포의 성질 중 상태 분할 개수(k)가 증가함에 따라 확률분포의 분산 값이 점점 작아져서 결국에는 unit impulse 값을 가지는 사실을 적용하여[22], 본 모델링에서 요구되는 재활 상태와 작업전이 상태에서의 확정값을 가지는 sojourn time 문제를 해결하였다. 그림 2의 모든 상태에서 입력과 출력 비율이 일정해지는 평형(steady-state) 일 때의 균형 방정식(balance equation)은 다음과 같다.

$$\lambda_u P_n=\mu P_{n-1}+ku_r P_{R_{n,k}}$$

$$(u+\lambda_u)P_i=\mu P_{i-1}+ku_r P_{R_{i,k}}+k\lambda_s P_{S_{i+1,k}}, \mathrm{i}=1,2,\ldots,\mathrm{n}\text{-}1$$

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

$$(\lambda+\lambda_r)P_{u_i}=\lambda_u P_i \quad , \mathrm{i}=1,2,\ldots,\mathrm{n}$$

$$k\mu_r P_{R_{i,j}}=\lambda_r P_{u_i}, \mathrm{i}=1,2,\ldots,\mathrm{n}, \mathrm{j}=1,2,\ldots,\mathrm{k}$$

$$k\lambda_S P_{S_{i,j}}=\lambda P_{u_i}, \mathrm{i}=2,3,\ldots,\mathrm{n}, \mathrm{j}=1,2,\ldots,\mathrm{k}$$

$\sum_{i=0}^{n}P_i+\sum_{i=1}^{n}P_{u_i}+\sum_{i=1}^{n}\sum_{j=1}^{k}P_{R_{i,j}}+\sum_{i=2}^{n}\sum_{j=1}^{k}P_{S_{i,j}}=1$; conservation 방정식

각 상태 확률의 의미는 아래와 같다.

P_i : 시스템이 평형일 때, 정상 상태 i 에 머물 확률

P_{u_i} : 시스템이 평형일 때, 불안정 상태 i 에 머물 확률

$P_{R_{i,j}}$: 평형 시스템의 서버 개수가 i 이고 재활 작업에 들어갈 때, 추가로 거쳐야할 부분재활 상태의 개수가 k-j 개 남아있을 확률

$P_{S_{i,j}}$: 평형 시스템의 서버 개수가 i 일 때, 서버의 고장으로 인한 작업전이 시에, 추가로 거쳐야할 부분작업전이 상태의 개수가 k-j 개 남아있을 확률

위의 균형 방정식과 각 상태에서 머물 확률의 총합이 1이 되는 conservation 방정식을 결합한 연립 방정식을 풀면, 시스템이 평형일 때, 각 상태에 머물 확률을 얻을 수 있다.

$$P_n=\left[\begin{array}{l}\left(1+\frac{\lambda_u}{\lambda+\lambda_r}\left(1+\frac{\lambda_r}{u_r}+\frac{\lambda}{\lambda_s}\right)\right)\sum_{i=1}^{n}\left(\frac{\lambda}{u}\frac{\lambda_u}{\lambda+\lambda_r}\right)^{n-i} \\ +\frac{\lambda_\mu}{\lambda+\lambda_r}\left(\frac{\lambda}{u}-\frac{\lambda}{\lambda_S}\right)\left(\frac{\lambda}{u}\frac{\lambda_u}{\lambda+\lambda_r}\right)^{n-1}\end{array}\right]^{-1}$$

$$P_i=\left(\frac{\lambda}{u}\frac{\lambda_u}{\lambda+\lambda_r}\right)^{n-i}P_n \quad ,i=0,1,2,\ldots,n$$

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

$$P_{R_{i,j}}=\frac{\lambda_r}{ku_i}\frac{\lambda_u}{\lambda+\lambda_r}P_i \quad ,i=1,2,\ldots,n \quad ,j=1,2,\ldots,k$$

$$P_{S_{i,j}}=\frac{\lambda}{k\lambda_s}\frac{\lambda_u}{\lambda+\lambda_r}P_i \quad ,i=2,3,\ldots,n \quad ,j=1,2,\ldots,k$$

즉 시스템 운영 파라미터를 이용하여 P_n을 계산하면, 그 외의 모든 정상 상태(P_i), 불안정 상태(P_{u_i}), 재활 상태($P_{R_{i,j}}$) 및 작업전이 상태($P_{S_{i,j}}$)에서의 확률을 차례로 계산할 수 있다. 그림 4는 서버의 중복도를 2로 한 cold standby 이중계(duplex) 클러스터 시스템 운영 상태 모델의 예이며 시스템의 가동이 평형 상태일 때의 확률은 다음과 같다.

1) 정상 상태 :

$$P_2+P_1=\left(1+\frac{\lambda}{u}\frac{\lambda_u}{\lambda+\lambda_r}\right)P_2=\left(1+\frac{\lambda}{u}\frac{\lambda_u}{\lambda+\lambda_r}\right)\left[\begin{array}{l}\left(1+\frac{\lambda_u}{\lambda+\lambda_r}\left(1+\frac{\lambda_r}{u_r}+\frac{\lambda}{\lambda_s}\right)\right)\sum_{i=1}^{2}\left(\frac{\lambda}{u}\frac{\lambda_u}{\lambda+\lambda_r}\right)^{2-i}\\ +\frac{\lambda_u}{\lambda+\lambda_r}\left(\frac{\lambda}{u}-\frac{\lambda}{\lambda_s}\right)\left(\frac{\lambda}{u}\frac{\lambda_u}{\lambda+\lambda_r}\right)\end{array}\right]^{-1}$$

2) 불안정 상태 :

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

3) 재활 상태:

$$\sum_{j=1}^{k}\left(P_{R_{2,j}}+P_{R_{1,j}}\right)=\frac{\lambda_r}{u_r}\frac{\lambda_u}{\lambda+\lambda_r}P_2+\frac{\lambda_r}{u_r}\frac{\lambda_u}{\lambda+\lambda_r}P_1$$

4) 작업전이 상태:

$$\sum_{j=1}^{k}P_{S_{2,j}}=\frac{\lambda}{\lambda_s}\frac{\lambda_u}{\lambda+\lambda_r}P_2$$

5) 고장 상태 :

$$P_0=\left(\frac{\lambda}{u}\frac{\lambda_u}{\lambda+\lambda_r}\right)^2P_2$$

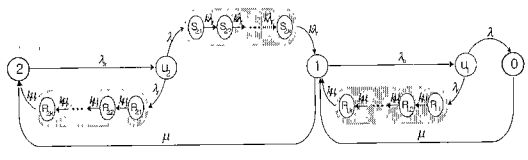

그림 4 cold standby 이중계 클러스터 시스템의 상태 전이도

불안정 상태에서 가동되고 있는 서버(u_i)는, λ의 변화율로 하드웨어적인 고장이 발생하며, 서버 두 대의 가동이 모두 중지된 고장 상태(0)에서는 μ의 변화율로 고장이 수리된다. 서버의 장시간 가동으로 인한 소프트웨어 노화로 인해 서버의 성능이 저하되는 불안정 상태에 있을 경우, 고의로 시스템의 가동을 멈추는 재활 상태($R_{2,j}$, $R_{1,j}$)로 가거나 혹은 고장이 발생하게 된다. 주서버의 고장으로 인해 발생한 여분서버로의 작업전이는 작업전이 상태($S_{2,j}$)를 통해 이루어진다. 서버가 한 대인 단일계(simplex) 시스템의 경우에는 여분서버가 없으므로, 작업전이 상태가 모델링에 포함되지 않는다

4. 실험 및 성능 평가

인터넷 기반 고가용도 cold standby 클러스터 시스템 서비스의 특성에 적합한 소프트웨어 재활 정책을 개발하기 위하여, 재활률, 고장률, 수리률, 가동 시간 및 손실 비용 등의 변수를 복합적으로 고려하였다. 서버의 가동이 중지되는 downtime의 절대적 크기보다는, 이로 인해 발생하는 손실 비용이 더욱 중요한 요소임을 고려하여 가용도 및 손실 비용을 계산하였다.

① 가용도

소프트웨어 재활 기법을 적용한 cold standby 클러스터 시스템의 가용도(AvailCss)는 모든 서버의 동작이 멈춘 고장 상태(P_0) 및 시스템 서비스가 일시적으로 중지되는 재활 상태 및 작업전이 상태에서 머물 확률을 제외함으로써 계산된다.

$$\text{AvailCss}=1-\left(P_0+\sum_{i=1}^{n}\sum_{j=1}^{k}P_{R_{i,j}}+\sum_{i=2}^{n}\sum_{j=1}^{k}P_{S_{i,j}}\right)$$

② 손실 비용

클러스터 시스템의 갑작스런 가동 정지로 인한 단위 시간당 손실 비용을 C_f, 재활 작업으로 인한 단위 시간당 손실 비용을 C_r, 작업 전이로 인한 단위 시간당 손실 비용을 C_s라 할 경우, 일반적으로 예상 가능한 시스템 정지 비용은 불시 정지로 인한 손실 비용에 비해 훨씬 저렴하게 들므로, $C_f >> C_r$ 의 관계가 성립한다. cold standby 클러스터 시스템의 가동 시간에 대한 손실 비용(CostCss)은 서버의 가동시간(T)의 함수로 아래와 같이 정의된다.

$$\text{CostCss}(T)=\left[P_0\times C_f+\left(\sum_{i=1}^{n}\sum_{j=1}^{k}P_{R_{i,j}}\right)\times C_r+\left(\sum_{i=2}^{n}\sum_{j=1}^{k}P_{S_{i,j}}\right)\times C_s\right]\times T$$

실험에 사용된 cold standby 이중계 클러스터 시스템의 운영 파라미터는 표 1과 같다[21]. 클러스터 시스템의 연속 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 평균 10분($1/\mu_r$)이 소요된다. 주서버의 고장으로 인해 발생하는 여

표 1 시스템 운영 파라미터[21]

시스템 운영 방식(n)	2 (이중계)
연속 가동기간(T)	1년
서버 고장률(λ)	1회/년
서버 수리률(μ)	1회/시간
서버 불안정률(λ_u)	1회/15일
재활률(λ_r)	1회/개월
재활작업 시간($1/\mu_r$)	10분
작업전이 시간($1/\lambda_s$)	3분
downtime 비용(C_f)	1000
rejuvenation 비용(C_r)	1
작업전이 비용(C_s)	100
분할 stage 수 (k)	10

분서버로의 작업 전이시간($1/\lambda_s$)은 평균 3분이 소요되며, 불시정지로 인해 발생하는 비용(C_f)은 재활 작업시 발생하는 비용(C_r)의 1000 배가되며, 작업전이 비용(C_s)은 불시정지 비용의 10% 이다.

고가용성 클러스터 시스템을 구성하고 있는 서버의 재활률, 고장률, 작업전이 시간 및 가동 대수(단일계, 이중계 및 다중계)에 따른 가용도 및 손실비용간의 관계를 파악하기 위하여 그래프를 작성하였다.

그림 5는 클러스터 시스템에서 소프트웨어 재활을 하지 않는 경우(재활률 0)와 재활 실시 간격을 일주일에서 6개월 사이로 했을 때, 시스템의 가용도를 변화 추세를 나타낸다. cold standby 시스템에서 재활을 자주 한다는 의미는 재활 작업으로 인한 서비스 중단을 의미하기 때문에 가용도가 재활률에 반비례하는 추세, 즉 재활을

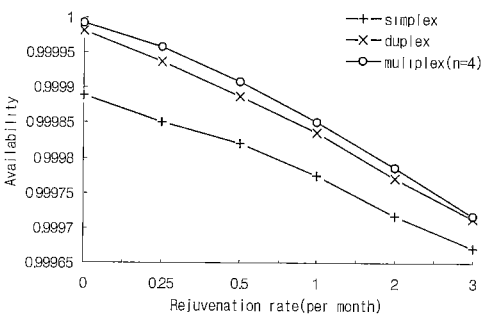

그림 5 재활률에 대한 가용도의 변화

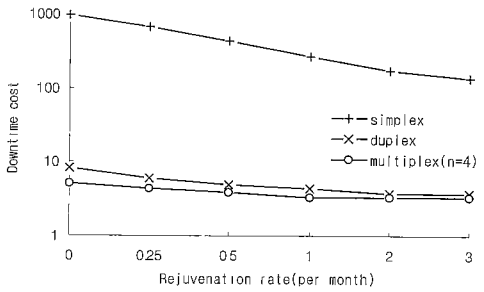

그림 6 재활률에 대한 손실비용의 변화

자주 할수록 가용도가 감소하는 현상을 나타냈다. 한편 한 대의 서버만이 가동되는 단일계 시스템의 가용도에 비해서 이중계 이상의 고가용성 클러스터 시스템의 가용도가 높게 나오는 것을 확인할 수 있으며, 이중계 이상의 서버 구성을 통해 개선될 수 있는 가용도의 상승폭은 상당히 미미함을 그래프에서 확인 할 수 있다. 이는 일반적으로 고가용도를 확보하기 위해 이중계 클러스터 시스템 구성이 가장 비용 효율적임을 나타낸다.

그림 6은 소프트웨어 재활 기법의 현실 문제 적용 가능성의 예를 보여주는 그래프로서, 재활 및 작업전이로 인해 발생하는 시스템 정지 비용에 비해, 불시의 시스템 정지로 인해 발생하는 비용이 클 경우, 소프트웨어 재활을 자주 하면 할수록 손실 비용이 모든 시스템 구성에서도 감소하는 추세를 나타내고 있다. 이 그래프에서 소프트웨어 재활을 수행할 경우, 비록 가용도 척도의 절대값은 떨어지지만, 현실적으로 시스템 운영자에게 더욱 중요한 성능 평가 요소인 손실비용 절감의 가능성을 제시하고 있다(그래프에서 손실비용의 효과적 표시를 위해 로그(log) 간격을 사용하였음).

서버의 하드웨어적인 신뢰도(고장 시간 간격)가 클러스터 시스템의 가용도와 손실비용에 미치는 영향을 그림 7과 8에 나타내었다. 단일계 시스템의 경우 1년 미만의 고장 시간 간격에 대해서는 가용도가 상당히 민감하게 변화는 것을 확인할 수 있으나, 이중계 이상의 클러스터 시스템을 구성할 경우, 하드웨어의 신뢰도가 시스템의 가용도에 미치는 영향의 변화 폭이 작음을 나타냈다. 이는 서론부에서도 밝힌 바와 같이, 하드웨어의 신뢰도보다는 소프트웨어의 신뢰도가 클러스터 시스템의 가용도 개선에 더 중요한 역할을 수행하고 있는 사례라 할 수 있다. 일반적으로 고장이 자주 발생하면 발생할수록(고장시간 간격이 짧을수록) 손실비용이 증가하는 것을 그림 8에서 확인할 수 있으며, 단일계 시스템의 손실

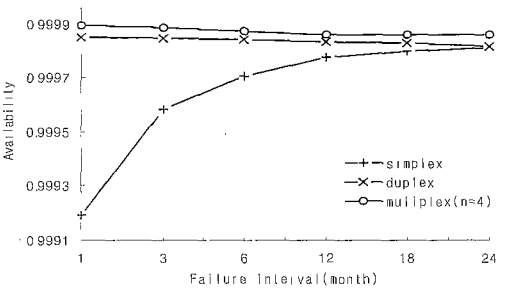

그림 7 고장 주기에 대한 가용도의 변화

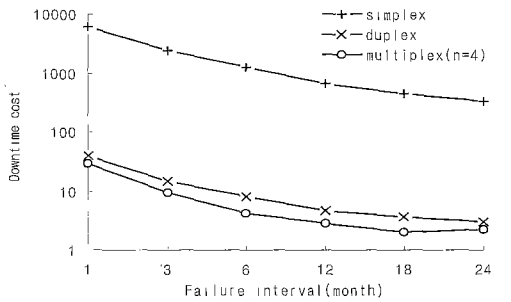

그림 8 고장 주기에 대한 손실비용의 변화

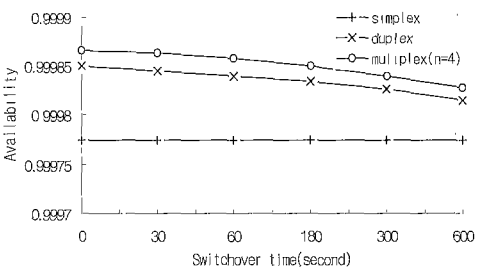

그림 9 작업전이 시간에 대한 가용도의 변화

비용은 이중계 시스템의 손실 비용에 비해 수 십배 차이가 나는 것을 확인하였다.

그림 9는 작업전이 시간이 클러스터 시스템에 미치는 영향을 나타낸다. 단일계 시스템의 경우 주서버만이 동작함으로 작업전이 시간과는 무관하게 항상 가용도가 일정하게 유지되며, 작업전이 시간을 0초(작업전이 상태가 포함되지 않을 때)에서 10분까지 변화 시켰을 때, 이중계 이상의 시스템 구성에서 가용도의 변화는 그리 크지 않음을 보이고 있다. 즉 클러스터 시스템을 소프트웨어 재활 정책을 사용하여 운영할 경우, 작업전이 시간이 시스템에 미치는 영향은 미미함을 알 수 있다. 또한 가용도 계산에 작업전이 시간이 포함되지 않은 기존 연구 결과[21]와 비교해보면, 작업전이 상태가 포함된 본 연구 논문의 가용도가 낮은 것을 확인할 수 있다.

재활작업 시간이 평균 10분이 소요되고 재활 상태의 분할 개수가 10일 때, 각 부분재활 상태의 평균재활 시간은 1분이 된다. 10개의 부분재활 시간을 모두 합한 10-stage Erlangian 분포의 평균은 지수 분포(k=1)와 동일한 10분이 되지만, 분산값은 지수 분포일 때(100분)의 10%인 10분으로 줄게 된다. 만약 k를 더욱 증가시킬 경우, k-stage Erlangian 분포의 분산값이 k의 비율로 줄게되어 원하는 확정시간을 가지는 재활작업 시간을 표현할 수 있게 된다.

5. 결 론

본 연구에서는 고가용도 cold standby 클러스터 시스템의 결함 예방 성능 분석을 위해, method-of-stages 개념을 적용하여, 특정 상태에서 머무는 시간을 지수분포들의 합으로 표현한 k-stage Erlangian 분포를 사용하였으며, 각 상태에서 머무는 시간을 분할하는 개수를 증가시킴으로서 머무는 시간을 확정적으로 표현하려고 시도하였다. 즉 본 논문에서는 고가용도 cold standby 클러스터 시스템의 운영 상태에 대한 상태 전이도에서, 임의의 상태에서 머무는 시간분포가 memoryless 성질을 만족하지 않아도 되는 semi-Markov 프로세스 문제를 해결하였다. 수학적 분석을 통해 구한 고가용성 cold standby 클러스터 시스템 재활 모델의 closed-form 해는 다양한 시스템 운영 상태에 대한 실험을 통해 검증하였으며, 소프트웨어 재활을 통한 예방적 결함허용 기법의 적용 가능성이 높다는 것을 확인하였다. 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 고가용성 부하분산 클러스터 시스템의 성능 평가 모델을 연구할 예정이고, 그밖에 소프트웨어 재활 기법을 적용한 시스템의 부하 및 가동능력(performability)을 동시에 고려할 수 있는 방안에 대한 연구가 요청된다.

참 고 문 헌

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

[2] H. Zhu, T. Yang, Q. Zheng, D. Watson, O. Ibarra

and T. Smith, "Adaptive Load Sharing for Clustered Digital Library Servers," Proceedings of the The Seventh IEEE International Symposium on High Performance Distributed Computing, July, 1998.
[3] 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.
[4] R. Buyya, *High Performance Cluster Computing Volume 1: Architectures and Systems.* p. 849, Prentice-Hall, 1999.
[5] 오수철, 정상화, "클러스터 시스템 기술 동향", 정보과학회지 제18권, 제3호, pp. 4-10, 2000.3.
[6] 유찬수, "리눅스 클러스터링", 정보과학회지 제18권, 제2호, pp. 33-39, 2000. 2.
[7] H. Levendel, "Software Dependability in Wireless Systems," Annual IEEE Workshop on Fault-Tolerant Parallel and Distributed Systems, San Juan, Puerto Rico, USA, April 16, 1999.
[8] 권세오, 김상식, 김동승, "리눅스 클러스터형 웹 서버 설계", 정보과학회지 제18권, 제3호, pp. 48-56, 2000. 3.
[9] G. F. Pister, "In Search of Cluster," Prentice-Hall, 1998.
[10] B. Johnson, *Design and Fault-Tolerant Analysis of Digital Systems.* p. 584, Addison-Wesley Publishing Company, 1989.
[11] 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.
[12] R. Jain, *The Art of Computer Systems Performance Analysis.* p. 685, John Wiley & Sons Inc., 1991.
[13] 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.
[14] 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.
[15] 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.
[16] 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.
[17] 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.
[18] K. Vo, Y. Wang, P. Chung, and Y. Huang, "Xept: A Software Instrumentation Method for Exception Handling," in Proc. Int. Symp. on Software Reliability Engineering, Nov. 1997.
[19] 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.
[20] 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.
[21] 박기진, 김성수, 김재훈, "소프트웨어 재활 기법을 적용한 다중계 시스템의 가용도 분석", 한국정보과학회 논문지(시스템및이론), 제27권, 제8호, pp. 730-740, 2000. 8.
[22] L. Kleinrock, *Queueing Systems Volume 1: Theory.* p. 417, John Wiley & Sons Inc., 1975.
[23] K. Trivedi, *Probability and Statistics with Reliability, Queueing, and Computer Science Applications.* p. 624, Prentice-Hall, 1982.

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

김 성 수
1982년 서강대학교 전자공학과(공학사). 1984년 서강대학교 전자공학과(공학석사). 1995년 Texas A&M University, 전산학과(공학박사). 1983년 ~ 1986년 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