소프트웨어 재활기법을 적용한 (n,k) 클러스터 시스템의 가용도 향상 모델

(Availability Improvement Model of (n,k) Cluster Systems using Software Rejuvenation)

이 재 성 † 박 기 진 †† 강 창 훈 ††† 박 범 주 †††† 김 성 수 †††††
(Chaesung Lee) (Kiejin Park) (Changhun Kang) (Bumju Park) (Sungsoo Kim)

요 약 인터넷 기반 시스템에서는 고가용도와 고성능을 제공해야하며, 클러스터 시스템 기술은 이에 대한 하나의 해결책으로 떠오르고 있다. 클러스터 시스템을 사용하는 중요한 목적은 성능과 가용도의 확보에 있으며, 고가용도 클러스터 시스템은 구성 노드들 중 일부에 결함이 발생했을 때 이를 비용·효율적으로 해결한다. 본 논문은 (n,k) 클러스터 시스템의 성능을 고려한 가용도 개선과 손실비용 분석에 관한 연구로 소프트웨어 재활 기법을 적용한 (n,k) 클러스터 시스템의 가용도 모델을 제안하였으며, 고가용도가 요청되는 시스템에서 소프트웨어 재활은 가용도 향상을 가져오는 유용한 기법 중의 하나임을 파악하였다.

키워드 : 결함 허용, 가용도, 클러스터, 소프트웨어 재활

Abstract Internet-based computer systems have to provide both high-availability and high-performance. Cluster technology has been used to obtain availability and performance simultaneously. Generally, high-availability cluster systems tolerate a failure of a cluster node and cost-effectively solve it. In this paper, we study availability and downtime cost of (n,k) cluster systems. By considering performance, we model state transition of (n,k) cluster systems and apply software rejuvenation technique to improve availability of the system. We find that software rejuvenation can be used to improve availability of (n,k) cluster systems.

Key words : Fault Tolerance, Availability, Cluster, Software Rejuvenation

1. 개 요

웹(www), 전자상거래(e-commerce) 같은 온라인(online) 서비스를 제공하는 시스템은 중단 없는 서비스, 즉 고가용도를 제공할 수 있어야 하며, 이는 전자상거래에서 서비스의 중단은 실질적인 거래를 원하는 소비자나 잠재적인 소비자를 놓치는 결과를 초래하기 때문에, 그에 따른 엄청난 손실비용이 발생한다. 이처럼 시스템의 다운으로 인한 손실비용은 계속 증가할 것이고 시스템 의존도 또한 높아질 것으로 전망된다[1]. 따라서 컴퓨터 시스템은 고가용도와 고성능에 대한 요구를 충족시킬 수 있어야 하며, 클러스터 시스템은 이에 대한 하나의 해결책으로 인정받고 있다. 클러스터 시스템은 개별적인 서버들이 서로 단일 시스템처럼 작동하는 독립 시스템들의 그룹을 말하며 현재 이에 대한 많은 연구가 이루어지고 있다[2].

본 논문의 대상이 되는 고가용도 클러스터 시스템은 n대의 주(primary)서버와 k대의 백업(backup)서버로 구성되어 있다(그림 1 참조). 부하분배기(load balancer)와 실선으로 연결된 것이 주서버이고 점선으로 연결된 것이 백업서버를 의미한다. 모든 사용자의 서비스 요청은 부하분배기로 들어가며, 부하분배기는 주서버의 작업부하

· 이 논문은 2003년도 두뇌한국21사업에 의하여 지원되었음.
· 본 연구는 한국과학재단 목적기초연구(R05-2003-000-10345-0)지원으로 수행되었음.
† 비 회 원 : 텔슨전자(주) 개발그룹 소프트웨어팀
ChaeSung_Lee@telson.co.kr
†† 정 회 원 : 안양대학교 소프트웨어학과 교수
kiejin@aycc.anyang.ac.kr
††† 비 회 원 : 극동정보대학 멀티미디어과 교수
chkang@keukdong.ac.kr
†††† 비 회 원 : 아주대학교 정보통신전문대학원
bumjoo@samsung.com
††††† 종신회원 : 아주대학교 정보통신전문대학원 교수
sskim@ajou.ac.kr
논문접수 : 2001년 5월 24일
심사완료 : 2003년 3월 7일

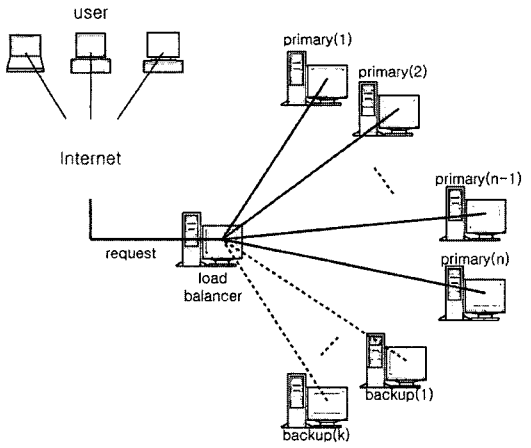


그림 1 (n,k) 클러스터 시스템 구조

(workload)를 균등하게 분할해 주는 역할을 한다. 주서버는 요청된 서비스의 수행을 하며, 백업서버는 주서버의 결함을 허용하기 위해 대기모드(standby) 상태로 존재한다. 클러스터 시스템에서 부하분배기가 반드시 필요한 것은 아니며 클러스터를 구성하는 서버들 중 어느 하나가 부하분배기의 역할을 대신 수행하기도 하며, 부하분배기는 본 논문의 분석대상에서 제외시켰다. 또한, 별도의 백업서버가 존재하지 않고, 주서버만으로 구성된 클러스터 시스템(N-way cluster system)도 있으며, 이 시스템에서는 임의의 주서버에 결함이 발생했을 때 결함이 없는 다른 주서버들이 결함이 발생한 서버의 일을 대신 수행함으로써 결함을 허용해 주지만, 백업서버를 가지는 방법보다 구현하기는 더 어렵다.

시스템의 다운은 크게 불시에 발생하는 중단과 계획된 중단 두 가지로 볼 수 있으며, 이 중 계획된 중단은 시스템의 유지, 보수를 위해서 필연적으로 해야 하는 것으로 철저한 관리체계를 두어 다운타임을 최소화시킬 수 있다. 계획되지 않은 중단은 일반적인 의미의 다운으로 위에서 언급한 엄청난 손실비용을 초래하는 주된 원인이며, 장애 원인으로는 크게 시스템 관리자의 실수, 소프트웨어의 결함, 하드웨어와 운영체제(OS)의 오류가 있으며, 이 가운데 소프트웨어의 결함으로 인한 시스템의 장애 비중이 점차 높아지고 있다[3,4]. 고가용도 클러스터 기술은 클러스터를 구성하는 서버들 중 임의의 서버가 다운되더라도 시스템이 정상적으로 가동될 수 있도록 해주는 기술로써 서버의 다운으로 인한 장애시간을 최소화시킬 수 있지만, 이 기술도 갑작스런 결함 발생을 예측할 수 없는 단점이 존재한다. 본 논문에서는 소프트웨어의 결함 발생을 사전에 막아주는 기술인 소프트웨어 재활 기법을 클러스터 시스템에 적용해서 이러한 단점을 보완하였다.

본 논문에서는 컴퓨터 시스템의 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. 즉 서버에서 가동되는 애플리케이션의 예방적 유지보수(preventive maintenance)를 통해서 계획되지 않은 중단을 최소화 시키는 개념이다. 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. 소프트웨어 재활 기법이 실제로 활용되 프전당시 패트리어트 미사일의 오작동 수정(Accumulated round-off Error) 3) IBM 사의 스스로 병을 고치는 서버(Self Healing)의 핵심 기술 등을 들 수 있으며, 패트리어트 미사일 제어시스템의 경우 일주일 이상 시스템을 계속해서 가동할 경우, 탄도 계산식에 오차가 누적되기 때문에, 이를 방지하기 위해 주기적으로 시스템을 재부팅하였다.

본 논문의 2장에서는 클러스터 시스템의 간략한 소개와 기존의 연구들에 대해 논하며, 3장에서는 소프트웨어 재활 기법을 적용한 (n,k) 클러스터 시스템(그림 1)의 가용도 모델 제안과 수학적 해석 방법으로 모델의 해를 구하였고, 4장에서는 수학적 해석 방법의 정확성을 검증하기 위해 다양한 시스템 파라미터를 가지고 실험을 수행하였다. 마지막으로 5장에서는 클러스터 시스템에서의 소프트웨어 재활에 대한 결론과 향후 연구에 대해 논하였다.

2. 관련 연구

클러스터 시스템을 사용하는 중요한 목적은 성능과 가용도의 확보에 있다. 일반적으로 가용도는 시스템이 일년 동안에 가동된 시간에 대한 평가척도로써 일년 동안 0.9999999의 가용도를 보이는 시스템을 FT(fault-tolerant) 시스템이라 하며, 이 경우 일년에 약 3초 이내의 다운시간(downtime)을 가진다. 일반적인 고가용도 시스템은 다운시간이 5분 내외인 0.99999의 가용도를 제공한다[5]. 이처럼 고가용도를 제공하는 클러스터 시스템을 고가용

도 클러스터 시스템이라 하며 클러스터에 있는 서버들 중 일부가 결함이 발생했을 때 이를 해결한다. 현재 고가용도 클러스터 시스템에 대한 연구가 활발히 이루어지고 있으며, HA(high availability) 프로젝트에서는 결함 허용을 제공하기 위한 기법으로 heartbeat을 연구하고 있다[6,7].

고가용도 시스템을 구성하기 위해서는 시스템의 결함을 미리 예측하여 이에 대비하는 것이 매우 중요하지만 현재 대부분의 시스템은 결함을 미리 예측하기보다는 결함이 발생한 후 복구하는데 걸리는 시간을 줄이는 것에 초점을 맞추고 있으며, 이를 위해 가장 많이 사용되고있는 방법이 바로 작업전이 기법이다. 작업전이 기법은 결함이 발생한 서버를 찾아서 그 서버에서 수행되던 작업을 다른 서버에서 대체 수행하기 위한 방법으로 이에 소요되는 시간을 작업전이 시간(switchover time)이라고 한다. 작업전이 시간의 단축을 통해서 가용도 향상을 가져올 수는 있지만, 결함 발생을 예측하지 못함으로써 다운시간 동안 발생하는 손실의 양을 파악하기가 어렵다는 단점이 있다.

본 논문에서 가용도 개선을 위해서 사용한 소프트웨어 재활 기법에 대한 자세한 내용은 [8,9,10,11,12,13,14,15,16,17,18,19,20,21]에 소개되어 있으며, 소프트웨어 재활 기법은 소프트웨어가 불안정한 상태에서 계속 실행되는 것을 막고, 컴퓨터를 처음 가동했을 때와 같은 안정적인 상태에서 다시 소프트웨어를 실행시킴으로써 미연에 결함 발생을 막아보려는 방법이다.

[8,9,10,11,12,13]에서는 소프트웨어 재활 개념에 대한 소개와 1대의 서버로 구성된 단일계(simplex) 시스템에서의 재활에 대한 연구를 논했으며, [14,15,16,17,18,19]에서는 소프트웨어적 결함의 원인이 되는 소프트웨어 노화(software aging) 현상에 대해 집중적으로 다루었다. 노화 현상은 애플리케이션이 계속해서 실행됨으로 인해서 점차 그 성능이 감소하는 현상을 의미한다.[20,21,22]은 백업서버를 갖는 다중계(multiplex) 시스템에서의 소프트웨어 재활에 대해 언급하였으며, 특히 [20]은 백업서버를 사용한 최초의 논문으로 백업서버의 운영에 대한 전략을 cold standby와 hot standby로 구분하였다. [21,22]에서는 [20]에서 고려하지 않은 작업전이 시간을 고려한 논문으로 [21]은 hot standby를 백업전략을 분석하였으며, [22]는 cold standby 백업전략 사용하여, 작업 전이 시간이 재활에 미치는 영향에 대한 분석을 수행하였다. 하지만 [20,21,22]에서는 주서버의 수를 한 대로 고정(n=1)하였다.

본 논문에서는 주서버의 수가 2대 이상으로 구성된 보다 일반적인 클러스터 시스템((n,k)-way)을 분석하였다. 가동되는 주서버의 수가 변할 경우에 전체 시스템의 가용도를 서버의 수와 무관하게 평가하는 기존 방법에는 무리가 있으며, 이를 해결하기 위해 성능평가척도(가동되고 있는 주 서버의 수)를 고려한 가용도 연구가 필요하다[23]. 여러 대의 주서버로 구성된 클러스터 시스템에서는 가동되는 서버의 수에 의해서 시스템의 처리 용량은 변하게 되며, 주서버의 결함은 전체 시스템의 성능에 영향을 주게 된다. 따라서 본 논문에서는 성능을 가용도 정의에 포함하였고, 일정 수준이상의 가용도와 성능을 유지하기 위해 필요한 백업서버의 수를 구하였다.

3. 시스템 모델

본 논문에서 제안한 (n,k) 클러스터 시스템 모델은 그림 2와 같으며, 사용된 가정들은 다음과 같다.

- (n,k) 클러스터 시스템에서 $n \geq 2$ 이다(n=1인 경우 [19,20] 참조).
- (n,k) 클러스터 시스템에서 각 서버의 고장률(λ)은 동일하다.

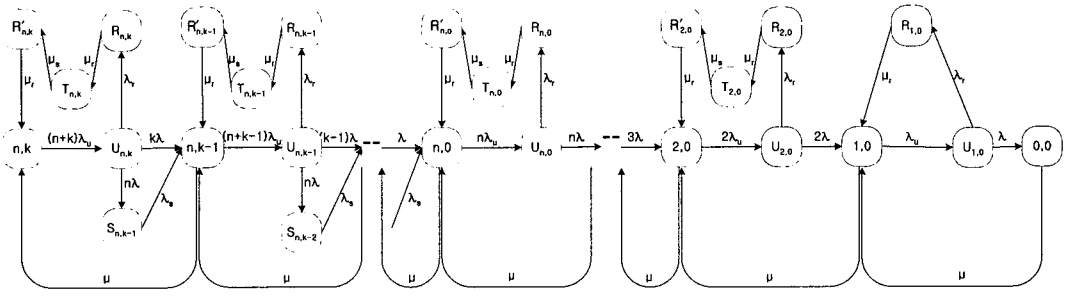
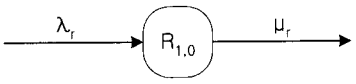


그림 2 (n,k) 클러스터 시스템의 가용도 모델

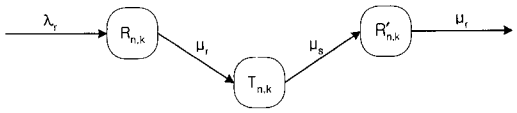
· 고장난 서버를 수리하는 수리률(μ)은 동일하다.
· 서버의 가동을 주기적으로 멈추는 재활률(λ_r)은 모든 가동 상태에서 동일하다.
· 장시간 가동으로 인한 서버의 불안정률(λ_u)은 모든 가동 상태에서 동일하다.
· 재활작업 시간($1/\mu_r$)은 동일하며 서버 수에 무관하다.
· 주서버에서 백업서버로의 작업전이 시간($1/\lambda_s$)은 동일하며 서버 수에 무관하다.
· 재활작업 중 발생하는 재활 대상 서버의 스위치 시간($1/\mu_s$)은 동일하며 서버 수에 무관하다.
· 재활에 들어갈 경우, 모든 서버를 재활하는데 중단 없는 서비스를 제공하기 위해 재활대상이 되는 서버들을 두 개의 그룹으로 나누어서 한 번에 한 개의 그룹에 속한 서버들만 재활을 수행하며 다른 하나의 그룹은 계속적으로 서비스를 수행한다.
· 백업서버의 재활은 두 번에 걸친 재활 과정 중 어느 한쪽에서 한번에 재활을 해준다.
· 현재 가동되고 있는 모든 서버들이 재활 대상이다.
· 모든 상태에서 머무는 시간은 지수분포를 따른다.

본 모델은 총 n+k대의 동일한 서버들로 구성되어 있으며, 이 중 n대는 주서버로써 실제 서비스를 수행하는 서버들이고 나머지 k대는 대기중인 백업서버들이다. 가용도 모델에서 각각의 상태(state)를 5가지로 분류하였으며, 이것은 서버에서 결함이 발생할 확률이 거의 없는 정상 상태 $\{(n,k), (n,k-1), \cdots, (n,0), \cdots, (1,0)\}$, 서버의 장시간 가동에 의한 노화 현상으로 서버의 결함 발생 확률이 높아지는 불안정 상태 $\{U_{n,k}, U_{n,k-1}, \cdots, U_{n,0}, \cdots, U_{1,0}\}$, 서버의 재활을 수행하는 재활작업 상태$\{(R_{n,k}, \cdots, R_{1,0}), (R'_{n,k}, \cdots, R'_{2,0}), (T_{n,k}, \cdots, T_{2,0})\}$, 서버의 결함으로 주서버가 서비스가 불가능할 때 백업서버가 여분 서버를 대체하는 작업전이 상태 $\{S_{n,k-1}, S_{n,k-2}, \cdots, S_{n,0}\}$, 마지막으로 모든 서버들의 결함으로 서비스가 중지된 고장 상태 $\{(0,0)\}$이다. 그림 2의 가용도 모델에서는 정상 상태에서 시작해서 상태 전이률에 따라 해당되는 상태로 전이하게 된다. 예를 들면, 정상 상태(n,k)에서 일정 시간이 지난 후 불안정 상태($U_{n,k}$)로 이동하고, 불안정 상태에서는 재활을 하게 되면 재활작업 상태로 바뀐 뒤 정상 상태(n,k)가 되며, 고장이 발생하게 되면 주서버의 고장인지 백업서버의 고장인지를 판별한 후에, 주서버의 고장이면 백업서버로의 전이가 발생하는 작업전이 상태($S_{n,k-1}$)를 거친 후 백업서버가 한 대 줄어든 정상 상태(n,k-1)로 이동하고 백업서버의 고장일 경우에는 바로 정상 상태(n,k-1)로 전이하게 된다. 본 모델에서의 소프트웨어 재활 과정은 그림 3과 같이 이루어지며 이 과정 전체를 재활작업 상태로 표현하였다.

주서버가 1대일 경우((a) 참조)는 재활작업 과정이 한 번에 수행되며 전체 서버(백업서버 포함)가 동시에 재활에 들어간다. (b)의 경우에서 재활 대상인 n대의 주서버들은 재활작업에 들어갈 때 절반($\lfloor n/2 \rfloor$대)의 서버를 먼저 재활하고 난 후 나머지 절반($\lceil n/2 \rceil$대)의 서버가 재활작업을 수행하며, 재활작업을 끝낸 서버들과 재활작업에 들어갈 서버들간의 스위치 시간(재활작업을 끝 낸 서버가 서비스를 수행할 때까지 걸리는 시간)이 발생하게 된다. (b)에서 백업서버들은 서비스 수행과는 무관하기 때문에 모든 백업서버들의 재활작업은 첫 번째 또는 두 번째의 재활작업 과정 중 어느 한 곳에서 동시에 수행된다.



(a) 서버가 1대일 경우의 재활작업 과정



(b) 서버가 2대 이상일 경우의 재활작업 과정

그림 3 재활작업 과정

본 논문에서 사용된 파라미터들의 정의는 표 1과 같다.

그림 2의 (n,k) 클러스터 시스템의 상태 모델에 대한 평형상태 방정식은 다음과 같으며, 평형상태 방정식에서 사용된 첨자 i는 각각의 상태에서 결함이 발생하지 않은 서버의 수(n+k)를 의미한다. 예를 들면, i=n+k 라고 하면 모델에서 (n,k) 상태를 표현하며 i=n은 (n,0) 상태를 나타낸다.

$$(n+k)\cdot\lambda_u\cdot P_{n+k}=\mu_r\cdot P_{r'_{n+k}}+\mu\cdot P_{n+k-1}$$

$$(i\cdot\lambda_u+\mu)\cdot P_i=\mu_r\cdot P_{r'_i}+(i+1-n)\cdot\lambda\cdot P_{u_{i+1}}+\lambda_s\cdot P_{s_i}$$
$$+\mu\cdot P_{i-1},\quad i=n,n+1,\cdots,n+k-1$$

$$(i\cdot\lambda_u+\mu)\cdot P_i=\mu_r\cdot P_{r'_i}+(i+1)\cdot\lambda\cdot P_{u_{i+1}}+\mu\cdot P_{i-1},$$
$$i=1,2,\cdots,n-2,n-1$$

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

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

$$\mu_r\cdot P_{r_i}=\lambda_r\cdot P_{u_i},\quad i=1,2,\cdots,n+k$$

$$\mu_r\cdot P_{r'_i}=\mu_s\cdot P_{t_i},\quad i=2,3,\cdots,n+k$$

$$\mu_s\cdot P_{t_i}=\mu_r\cdot P_{r_i},\quad i=2,3,\cdots,n+k$$

$$\lambda_s \cdot P_{s_i} = n \cdot \lambda \cdot P_{u_{i+1}}, \qquad i = n, n+1, \cdots, n+k-1$$

$$\sum_{i=0}^{n+k} P_i + \sum_{i=1}^{n+k} (P_{u_i} + P_{r_i}) + \sum_{i=2}^{n+k} (P_{r'_i} + P_{t_i}) + \sum_{i=n}^{n+k-1} P_{s_i} = 1$$

위의 평형 상태 방정식과 각 상태에 머물 확률의 총합이 1이 되는 사실을 결합하여 연립방정식을 풀면 다음과 같은 closed-form 형태의 평형상태 확률을 얻을 수 있다.

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

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

$$P_{r'_i} = \frac{\lambda_r}{\mu_r} \cdot \frac{i \cdot \lambda_u}{\lambda_r + i \cdot \lambda} \cdot P_i, \; i = 2, 3, \cdots, n+k$$

$$P_{t_i} = \frac{\lambda_r}{\mu_s} \cdot \frac{i \cdot \lambda_u}{\lambda_r + i \cdot \lambda} \cdot P_i, \; i = 2, 3, \cdots, n+k$$

$$P_{s'_i} = \frac{n \cdot \mu}{(i+1) \cdot \lambda_s} \cdot P_i, \; i = n, n+1, \cdots, n+k-1$$

$$P_i = \left(\frac{\lambda \cdot \lambda_u}{\mu}\right)^{n+k-i} \cdot \prod_{l=0}^{n+k-1-i} \frac{(n+k-l)^2}{\lambda_r + (n+k-l) \cdot \lambda} \cdot P_{n+k}, \; i = 0, 1, \cdots, n+k-1$$

$$P_{n+k} = \left[1 + \sum_{i=0}^{n+k-1} \left\{ \left(\frac{\lambda \cdot \lambda_u}{\mu}\right)^{n+k-i} \cdot \left(\prod_{l=0}^{n+k-1-i} \frac{(n+k-l)^2}{\lambda_r + (n+k-l) \cdot \lambda} \right) \cdot \left(1 + \frac{\mu}{(i+1) \cdot \lambda} \cdot \left(1 + \frac{2 \cdot \lambda_r}{\mu_r} + \frac{\lambda_r}{\mu_s}\right)\right) \right\} + \sum_{i=0}^{k} \left\{ \left(\frac{\lambda \cdot \lambda_u}{\mu}\right)^{k-i} \cdot \left(\prod_{l=0}^{k-i} \frac{(n+k-l)^2}{\lambda_r + (n+k-l) \cdot \lambda} \right) \cdot \frac{\lambda_r + (n+i) \cdot \lambda}{(n+i)^2} \cdot \frac{n \cdot \mu}{(n+1+i) \cdot \lambda_s} - \left\{ \frac{n \cdot \mu}{(n+k+1) \cdot \lambda_s} + \left(\frac{\lambda \cdot \lambda_u}{\mu}\right)^{n+k} \cdot \left(\prod_{l=0}^{n+k-1} \frac{(n+k-l)^2}{\lambda_r + (n+k-l) \cdot \lambda} \right) \cdot \frac{\mu}{\lambda} \cdot \left(\frac{\lambda_r}{\mu_r} + \frac{\lambda_s}{\mu_s}\right) \right\} \right]$$

4. 실험 및 분석

그림 2의 가용도 모델을 이용하여 재활에 영향을 미치는 파라미터 값들의 변화에 대한 가용도와 손실비용 및 적절한 백업서버 수의 산출을 위해서, 본 논문에서는 시스템의 성능을 고려한 가용도와 손실비용을 분석하였다. 성능이 고려되지 않은 가용도와 손실비용에 대한 정의는 다음과 같다.

① 성능이 고려되지 않은 가용도 : 시스템을 구성하는 서버들 중 어느 하나라도 가동중일 경우 가용한 것으로 판단한다.

$$availability = 1 - \left(\sum_{i=2}^{n+k} P_{t_i} + P_{r_1} + P_0 \right)$$

② 성능이 고려되지 않은 손실비용 : 시스템의 연속 가동기간(T)에 대한 함수로써, 시스템이 서비스를 제공하지 못할 경우 발생하는 비용을 말하며, 성능이 고려되지 않은 가용도와 관련 비용을 결합하여 계산된다.

표 1 파라미터 정의

파라미터	정 의
n	주서버 수
k	백업서버 수
i or (n,k)	가동할 수 있는 서버 수(1, 2, ... , n+k)
U_i	계속적인 서버의 가동으로 인한 불안정한 상태
R_i	첫 번째 재활작업 상태
R'_i	두 번째 재활작업 상태
T_i	재활 대상 서버의 스위치 상태
S_i	고장난 서버의 작업전이 상태
P_i	i 상태에 머물 확률
P_{u_i}	U_i 상태에 머물 확률
P_{r_i}	R_i 상태에 머물 확률
$P_{r'_i}$	R'_i 상태에 머물 확률
P_{t_i}	T_i 상태에 머물 확률
P_{s_i}	S_i 상태에 머물 확률
C_f	모든 서버들의 고장으로 인한 단위시간당 손실비용
C_r	R_1 상태에서 일시적인 서비스 중지로 인한 단위시간당 손실비용
C_t	T_i 상태에서 일시적인 서비스 중지로 인한 단위시간당 손실비용
C_d	i, U_i 상태에서 성능감소로 인한 단위시간당 손실비용
C_{dr}	R_i, R'_i 상태에서 성능감소로 인한 단위시간당 손실비용
C_{ds}	S_i 상태에서 성능감소로 인한 단위시간당 손실비용

$$downtime\cos t = \left(\sum_{i=2}^{n+k} P_{t_i} \cdot C_t + P_{r_1} \cdot C_r + P_0 \cdot C_f \right) \cdot T$$

백업서버가 없는 클러스터 시스템에서 어느 하나의 서버에 결함이 발생했다면, 전체 클러스터 시스템의 처리 용량은 작아지지만 백업서버가 있을 경우에는 주서버에 결함이 발생하더라도 백업서버가 주서버를 대체하기 때문에 전체 클러스터 시스템의 처리 용량은 변하지 않게 된다. (n,k) 클러스터 시스템에서 n대가 정상적으로 서비스를 수행할 때 시스템의 성능이 100%라고 정의하였고, 서버 한 대의 결함으로 인해 성능은 1/n 만큼 감소한다고 가정하였다. 성능을 고려한 가용도와 손실비용에 대한 정의는 다음과 같다.

③ 성능을 고려한 가용도 : 100%의 성능으로 가동되

는 가용도를 나타낸다.

$$availability_p = \sum_{i=n}^{n+k} (P_i + P_{u_i})$$

④ 성능을 고려한 손실비용 : 100%의 성능으로 가동되지 못함으로 인해 발생하는 손실비용이다.

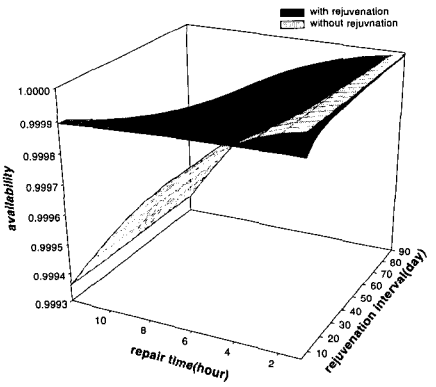
$$downtime\ \cos t_p = \left(\begin{array}{c} \sum_{i=1}^{n-1} \{(P_i + P_{u_i}) \cdot (n-1) \cdot C_d\} \\ + \sum_{i=2}^{n+k} P_{r_i} \cdot \left\lfloor \frac{n+k-i}{2} \right\rfloor \cdot C_{dr} \\ + \sum_{i=2}^{n+k} P_{r'_i} \cdot \left\lceil \frac{n+k-i}{2} \right\rceil \cdot C_{dr} \\ + \sum_{i=n}^{n+k-1} P_{s_i} \cdot C_{ds} + \sum_{i=2}^{n+k} P_{t_i} \cdot C_t \\ + P_{r_1} \cdot C_r + P_0 \cdot C_f \end{array} \right) \cdot T$$

표 2에 (n,k) 클러스터 시스템 모델의 실험에 사용된 기본 파라미터 값들을 표시하였으며, 특별한 언급이 없는 경우 모든 실험에 이 값들을 사용하였다[11]. 성능을 고려한 가용도 및 손실비용에 대한 정의는 각각의 상태에서 가동되는 서버의 수에 의해 영향을 받기 때문에, 단위시간당 재활로 인한 성능감소 손실비용(C_{dr}), 단위시간당 정상 상태와 불안정 상태에서의 성능감소 손실비용 간당 작업전이로 인한 성능감소 손실비용(C_{ds}), 단위시간(C_d)은 의미상으로 구분을 하기 위해서 사용될 뿐 성능에는 영향을 미치지 않는다. 파라미터 값들 중에서 C_d, C_{dr}, C_{ds}값들은 모두 같다고 가정하였다. 시스템의 재활작업 시간($1/\mu_r$)은 10분이 걸린다고 하였고, 재활작업 중 발생하는 재활서버들의 전이시간($1/\mu_s$)은 1분이 소요되며, 이는 재활작업이 계획된 것이고 작업전이 동안에 발생하는 결함 감지와 같은 부가적인 작업을 수행하지 않기 때문에 작업전이 시간보다는 적은 시간이 걸린다고 하였다. 따라서 전체 재활작업 과정을 수행하기 위한 시간은 최대 21분에서 최소 10분이 필요하다(그림 3참조). 작업전이 시간($1/\lambda_s$)은 백업서버가 대기모드 상태로 존재하기 때문에 재활작업 시간보다 적은 3분으로 가정하였다. 서버의 고장은 3달에 한번 발생하며, 재활은 1달에 한번 수행한다.

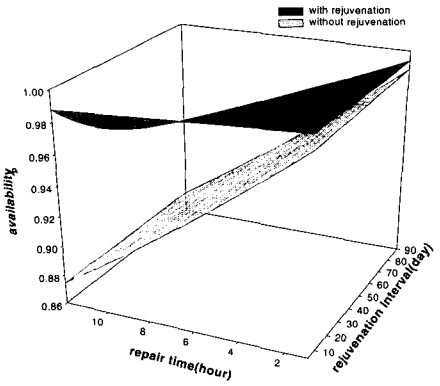
표 2 시스템의 기본 파라미터 값[11]

시스템의 기본 파라미터	값
주서버 수(n)	2대
서버의 고장주기	3달
불안정 상태 전이주기	7일
서버의 수리시간	12시간
재활작업 시간	10분
작업전이 시간	3분
재활 서버들간의 전이시간	1분
서버의 재활주기	30일
단위시간당 고장 발생 손실비용 (C_f)	1000
작업전이에 의한 서비스 중지로 발생하는 단위시간당 손실비용 (C_s)	100
재활작업에 의한 서비스 중지로 발생하는 단위시간당 손실비용 (C_s)	1
재활대상 서버의 전환에 의한 서비스 중지로 발생하는 단위시간당 손실비용 (C_t)	100
단위시간당 성능감소 손실비용(C_d, C_{dr}, C_{ds})	10
시스템의 연속 가동기간 (T)	1년

그림 4의 (a)는 수리시간과 재활주기에 따른 성능을 고려하지 않은 가용도의 변화를 보여준다. 수리시간이 짧을수록 재활 유·무에 상관없이 가용도가 모두 증가했으며, 수리시간에 대한 가용도의 변화는 재활을 했을 경우에 하지 않았을 경우보다 영향을 적게 받는다. 또한 수리시간과 재활주기 모두 길어지면서 가용도가 현저하게 낮아지는 현상을 보였다. 이는 재활주기가 길어지게 되면 재활을 하지 않는 것과의 차이가 줄어들기 때문에, 가용도가 낮아지는 것으로 판단된다. (b)는 시스템 성능이 100%일 경우, 수리시간과 재활주기에 따른 성능을



(a) 성능을 고려하지 않을 때



(b) 성능을 고려했을 때(성능 100%)

그림 4 재활주기와 수리시간에 따른 가용도의 변화

고려한 가용도의 변화를 나타내고 있으며, 재활을 했을 때 모든 서버(n대의 서버)가 정상적으로 가동되는 시간이 늘어남(가용도가 높음)을 알 수 있다. (a),(b)를 동시에 비교하면, 일반적인 가용도((a) 참조)는 거의 고가용도의 기준인 0.99999에 근접한 값을 갖지만, 성능을 고려한 가용도((b) 참조)에서는 이 값과 많은 차이를 보이고 있으며 이것은 결함을 허용하고 성능을 유지하기 위해 백업서버의 필요성을 보여준다. 한편 재활의 유·무와 관련해서는 (a),(b) 모두 재활을 하는 것이 좋고, 특히 (b)에서는 재활 유·무에 따른 가용도의 차이가 (a)보다 훨씬 크게 나타났다. 이는 소프트웨어 재활을 하는 것이 재활을 하지 않는 것보다 가용도면에서도 유리하고 성능 면에서도 이점을 얻을 수 있다는 것을 의미한다.

그림 5는 수리시간과 재활주기에 따른 손실비용의 변화를 나타낸다. 손실비용은 재활주기보다는 수리시간에 더 많은 영향을 받으며, 그림 4의 (a)의 가용도 그래프와 같이 비교하면, 가용도가 높을수록 손실비용은 낮아지며 그 반대의 경우도 성립한다. 또한 수리시간이 짧을 경우에는 재활을 하지 않는 것이 가용도가 더 높게 나타남에도 불구하고 손실비용은 오히려 재활을 했을 경우보다 더 높음을 알 수 있다. 이는 서버의 갑작스런 서비스 중단으로 인한 손실비용이 재활과 같은 계획된 중단으로 인해 발생한 것보다 더 크다는 것을 의미한다. 갑작스런 중단은 언제 어떤 상황에서 일어날지 모르기 때문에 손실비용이 커지는 반면에, 소프트웨어 재활은 재활정책에 의해서 계획적으로 수행되어지기 때문에 손실비용이 적게 나타나는 것이다.

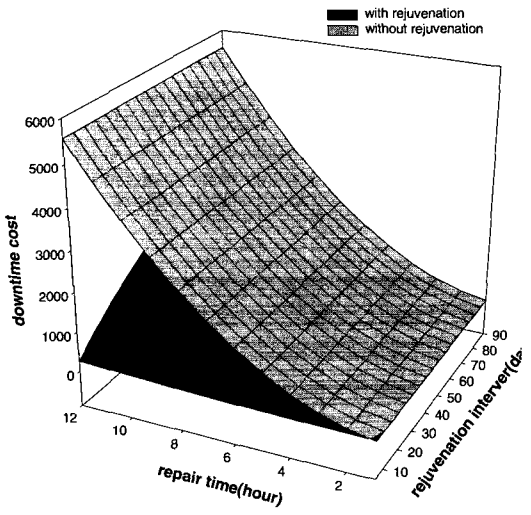


그림 5 수리시간과 재활주기에 따른 손실비용의 변화

그림 6은 일정 수준의 성능을 유지하기 위해 필요한 주서버 수에 따른 백업서버의 수를 보여주며, 이것은 일정 수준 이상의 가용도와 100% 성능을 유지하기 위해 필요한 최소한의 백업서버의 수를 의미한다. 예를 들면, 소프트웨어 재활을 하고 주서버가 4대일 경우, 0.99의 가용도와 100% 성능을 유지하기 위해 2대의 백업서버가 필요하다. 본 실험에서 백업서버의 수를 이러한 방식으로 구한 이유는 시스템의 성능을 고려하지 않은 가용도만을 가지고 백업서버의 수를 구하는 것은 어려우며, 백업서버가 주서버의 일을 대신 함으로써 가용도를 높일 뿐만 아니라 전체 시스템의 성능이 떨어지는 것을 막아줄 수 있기 때문이다. 한편 주서버의 수가 일정할 때,

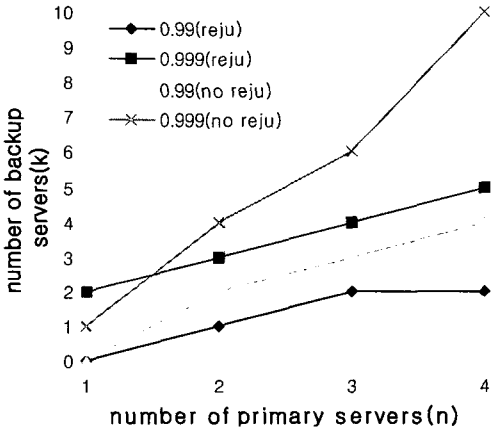


그림 6 일정 수준 성능을 유지하기 위한 백업서버 수의 변화

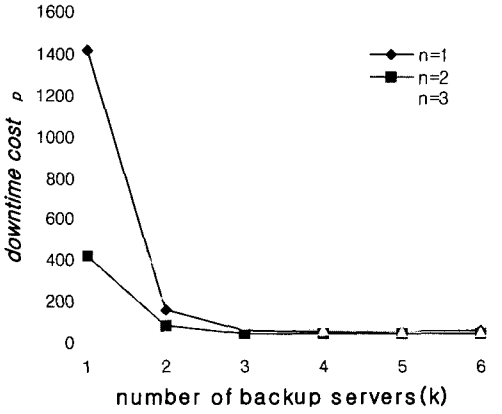


그림 7 백업서버 수에 따른 성능감소 손실비용 변화

재활을 하지 않았을 경우에는 백업서버의 수가 증가할수록 가용도 또한 증가하고 손실비용은 감소하는 경향을 보였지만, 재활을 했을 경우에는 가용도와 손실비용 모두 계속적인 증가와 감소 추세를 보이지 않았다(그림 7 참조). 이 실험을 통해서 재활을 하는 것이 가용도와 성능을 유지하기 위해서 보다 적은 수의 백업서버를 필요로 하며, 일반적으로 주서버의 수가 증가할수록 백업서버의 수 또한 증가함을 알 수 있다.

그림 8은 주서버가 10대일 경우 백업서버 수에 따른 성능분포별 가용도를 보여주고 있다. 성능이 70%일 경우의 가용도는 10대의 주서버 중 7대 이상이 가동된 가용도를 의미한다. 재활을 했을 경우에는 백업서버 수가 증가하면서 각각의 성능에 따른 가용도 차이가 차츰 작아지지만 재활을 하지 않았을 경우에는 어느 정도의 차이를 유지한다. 소프트웨어 재활을 함으로써 더 높은 가용도를 얻을 수 있을 뿐만 아니라 일정 수준의 성능과 가용도를 유지하기 위해서 더 적은 백업서버를 필요로 함을 알 수 있다.

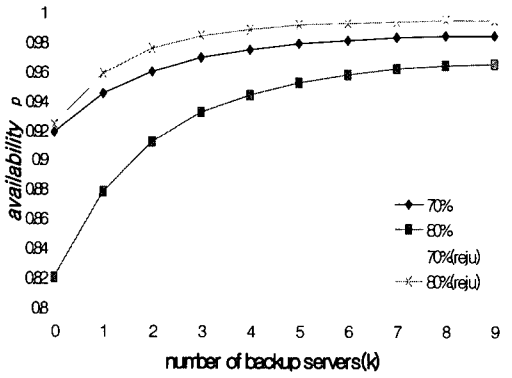


그림 8 백업서버 수에 따른 성능 분포별 가용도 변화 (n=10)

5. 결 론

본 논문에서는 (n,k) 클러스터 시스템의 성능을 고려한 가용도 개선과 손실비용 분석을 하였다. 클러스터 시스템의 가용도를 높이기 위해서 소프트웨어 재활 기법을 적용하였고, 소프트웨어 재활로 클러스터 시스템의 가용도가 증가함을 알 수 있었다. 일정 수준 이상의 가용도로 원하는 성능을 유지하기 위해 필요한 백업서버의 수를 구할 때도 재활을 함으로써 더 적은 수의 백업서버가 필요함을 파악하였다. 본 논문을 통해서 소프트웨어 재활은 고가용도를 요구하는 시스템에서 가용도 향상을 가져오는 유용한 기법 중의 하나임을 알 수 있었다. 마지막으로 본 논문에서는 시스템의 성능이 일정하게 감소하는 것으로 표현하였으나 앞으로 이를 개선하기 위한 연구를 수행할 예정이다.

참 고 문 헌

[1] 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.

[2] G. Pfister. In Search of Clusters: The Coming Battle in Lowly Parallel Computing. Prentice-Hall, NJ 1995, ISBN 0134376250.

[3] J. Gray and D. Siewiorek, "High-Availability Computer Systems," IEEE Computer, Vol. 24, No. 9, pp. 39-48, September 1991.

[4] 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, Vol. 21, No, 6, pp. 2-9, June 1991.

[5] Enterprise Computing, http://www.enterpriseweb.co.kr

[6] 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, Vol. 6, No. 10, pp.180-187, October 24-27, 1995.

[7] J. Han, H. Sun and H, Levendel, "Availability Requirement for Fault Management Server," Proceedings of the 25th Annual International Computer Software and Applications Conference, Vol. 25, No. 10, pp. 8-12, October 2001.

[8] 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, Vol. 1, No. 12, pp. 22-25, December 1995.

[9] 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, Vol. 6, No. 10, pp. 180-187, October 1995.

[10] S. Garg, A. Puliafito, M. Telek and K. Trivedi, "On the Analysis of Software Rejuvenation Policies," Annual Conference on Computer Assurance (COMPASS), Vol. 12, No 6, pp. 16-20, June 1997.

[11] S. Garg, A. Puliafito, M. Telek and K. Trivedi, "Analysis of Preventive Maintenance in Transactions Based Software Systems," IEEE Tran-

sactions on Computers, Vol. 47, No. 1, pp. 96-107, January 1998.
[12] Y. Huang, C. Kintala, N. Kolettis and N. Fulton, "Software Rejuvenation: Analysis, Module and Applications," Proceedings of 25th IEEE Fault-Tolerant Computing Symposium, Vol. 25, No. 6, pp. 381-390, June 1995.
[13] 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.
[14] A. Pfening, S. Garg, A. Puliafito, M. Telek and K. Trivedi, "Optimal Rejuvenation for Tolerating Soft Failures," 27th & 28th Performance Evaluation, Vol. 27-28, No, 10, pp. 491-506, October 1996.
[15] Y. Wang, Y. Huang, K. Vo, P. Chung and C. Kintala, "Checkpointing and Its Applications," Proceedings of 25th IEEE Fault-Tolerant Computing Symposium, Vol. 25, No. 1, pp. 22-31, June 1995.
[16] 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, Vol. 9, No. 11, pp. 282-292, November 1998.
[17] 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, Vol. 5, No.1, pp. 3-12, January 1986.
[18] E. Marshall, "Fatal Error: How Patriot Overlooked a Scud," Science, p. 1347, March 1992.
[19] 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, Vol. 3, No 11. pp. 26-27, February 1997.
[20] 박기진, 김성수, 김재훈, "소프트웨어 재활 기법을 적용한 다중계 시스템의 가용도 분석," 한국정보과학회논문지(시스템및이론), 제27권, 제8호, pp. 730-740, 2000. 8.
[21] 박기진, 김성수, "고가용도 Cold Standby 클러스터 시스템 성능 분석," 한국정보과학회논문지(시스템및이론), 제28권, 제3·4호, pp. 173-180, 2001. 4.
[22] 이재성, 박기진, 김성수, "소프트웨어 재활기법에 기반한 주-여분 서버 시스템의 작업전이 시간분석," 한국정보처리학회논문지, 한국정보처리학회, 제8-A권, 제2호, pp. 81-90, 2001. 6.
[23] V. Mainkar, "Availability Analysis of Transaction Processing Systems Based on User-Perceived Performance," 16th Symposium on Reliable Distributed Systems, Vol. 16, No. 10, pp. 10-16, October 1997.

이 재 성

1999년 아주대학교 정보및컴퓨터공학부(공학사), 2001년 아주대학교 정보통신전문대학원(공학석사), 2001년~현재 텔슨전자㈜ 개발그룹 소프트웨어팀, 관심 분야: 클러스터시스템, 성능분석, 결함허용

박 기 진

1989년 한양대학교 산업공학과(공학사). 1991년 포항공과대학교 산업공학과(공학석사). 91년~1996년 삼성종합기술원 기반기술연구소 전임연구원. 1996년~1997년 삼성전자(주) 소프트웨어센터 선임연구원. 1997~2001년 아주대학교 컴퓨터공학과 (공학박사). 2001~2002년 한국전자통신연구원 네트워크장비시험센터 선임연구원. 2002년~현재 안양대학교 소프트웨어학과 교수. 관심분야 : 고가용성클러스터컴퓨팅, Grid Computing, Software Analysis, 멀티미디어시스템, 시뮬레이션

강 창 훈

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

박 범 주

1989년 서울대학교 조선공학과(공학사). 1992년 포항공과대학교 산업공학과(공학석사), 1992년~현재 : 삼성종합기술원,삼성전자 첨단기술연수소, 2002년~현재 아주대학교 정보통신전문대학원 박사과정, 관심분야 : 결함허용, 성능분석, 클러스터컴퓨팅, 소프트웨어재활, 멀티미디어시스템

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. 1993년~1995년 Texas Transportation Institute 연구원. 1993년~1995년 Texas A&M University, 전산학과, T.A. 1997년~1998년 한국정보처리학회, 한국정보과학회 논문지 편집위원, 1996년~현재 아주대학교 정보통신전문대학원 부교수, 관심분야 : Autonomic Computing, Cluster System, Grid Computing, High Availability