그리드 자원정보 서비스를 위한 최적화된 동적 모니터링 인터벌에 관한 연구☆

An Optimal and Dynamic Monitoring Interval for Grid Resource Information Services

김 혜 주* 허 의 남** 이 웅 재*** 박 형 우****
Hye-Ju Kim Eui-Nam Huh Woong-Jae Lee Hyoung-Woo Park

요 약

그리드 기술은 지리적으로 분산된 컴퓨팅 자원을 활용하기에 다양한 자원에 대한 정보를 관리하는 기술이 필요하며, 이는 자원 관리 시스템에서 자원을 모니터링하여 이루어진다. 자원 상태 정보가 실시간으로 변하여 주기적으로 모니터링을 해야 하므로 이에 따른 오버헤드가 예상된다. 본 논문에서는 CPU 상태 변화 정도를 관찰하여 시스템 상태를 파악하고 이를 모니터링 주기에 적용시키고자 한다. 즉 그리드 환경의 중요한 요소인 자원의 상태를 고려하여 자율적으로 자원의 상태에 따른 모니터링 주기를 조절하는 시스템을 모델링 하였다. 자원의 CPU 상태를 관찰해 CPU의 상태 변화율에 따라 자원의 상태 변화를 파악하여 모니터링 주기를 조절함으로써 모니터링에 의한 오버헤드가 감소하였음을 실험을 통해 증명하였다.

Abstract

Grid technology requires use of geographically distributed resources from multiple domains. Resource monitoring services or tools consisting sensors or agents will run on many systems to find static resource information (such as architecture vendor, OS name and version, MIPS rate, memory size, CPU capacity, disk size, and NIC information) and dynamic resource information (CPU usage, network usage(bandwidth, latency), memory usage, etc.). Thus monitoring itself may cause system overhead. This paper proposes the optimal monitoring interval to reduce the cost of monitoring services and the dynamic monitoring interval to measure monitoring events accurately. By employing two features, we find out unnecessary system overhead is significantly reduced and accuracy of events is still acquired

Keyword : Grid, monitoring, optimal, interval, dynamic interval, events

1. 서 론

인터넷 대역폭의 확장으로 인해 LAN이나 자체 인터페이스를 이용하는 클러스터링에서 벗어나 국가나 전세계를 포함하는, 여기에는 단순히 컴퓨터만 피어로 참여하는 것이 아니라 대용량 저장 장치, 다양한 고성능 연구 장비들이 포함되고 있는데, 이렇게 통합된 형태를 그리드라고 한다. 즉 그리드는 네트워크로 연결된 컴퓨팅 자원을 사용자가 개인용 컴퓨터를 이용하듯이 사용할 수 있다는 개념에서 출발한 네트워크로 연결된 가상의 슈퍼컴퓨터를 말한다[1].

이기종의 컴퓨팅 자원과 대용량 저장장치, 다양한 고성능 연구 장비를 통합하는 그리드 환경에서 다양하고 많은 자원들은 그리드 자원 관리 시스템에 의해 관리되는데, 크게 자원 브로커와 정보서비스, 자원관리자로 구성되어 있다. 자원 브로

* 준 회 원 : 서울여자대학교 컴퓨터공학과 졸업예정(석사)
judy012@swu.ac.kr(제 1저자)
** 종신회원 : 서울여자대학교 정보통신공학부 교수
huh@swu.ac.kr(책임저자)
*** 종신회원 : 서울여자대학교 정보통신공학부 교수
wjlee@swu.ac.kr(공동저자)
**** 정 회 원 : 한국과학기술정보연구원 그리드실장
hyoark@kisti.re.kr(공동저자)
☆ 이 연구는 2003학년도 서울여자대학교 교내 특별연구 지원사업으로 수행되었음

커는 자원의 발견, 할당, 모니터링, 예약을 작업 단위로 자원 관리를 위한 모든 기능을 수행하고, 정보 서비스는 자원에 위치와 상태에 대한 정보를 수집하고 유지해 필요에 따라 제공해주는 역할을 한다. 그리고 자원관리자는 개별 자원에 대한 관리 기능을 담당한다. 이처럼 자원을 수집하고 자원의 정보를 파악하고 자원의 상태를 알기 위해서는 모니터링이 필요하다. 또한 그리드 상에서의 모니터링 기술은 오류가 감지된 응용프로그램, 네트워크, 호스트의 오류 혹은 overload가 감지될 경우 능동적으로 자원관리가 가능하도록 한다.[2][3]

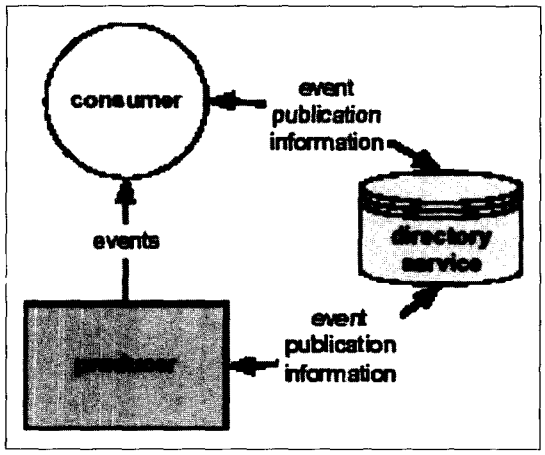
실시간으로 변하는 그리드 자원의 특성상 정확한 자원 정보를 파악하기 위해서는 모니터링 또한 실시간으로 이루어져야 할 것이다. 하지만 실시간으로 모니터링을 하는 것은 시스템 성능에 굉장한 오버헤드가 되므로 얼마나 효율적인 주기로 모니터링을 하는가가 중요하다. 예를 들어 자원 상태 변화가 거의 없는데 짧은 주기로 모니터링이 계속 이루어진다면 모니터링에 의한 오버헤드만이 증가할 것이며, 자원 상태 변화가 큰데 긴 주기로 모니터링이 이루어진다면 정확하지 않은 자원 정보를 가지고 자원 관리 시스템에서 자원 관리를 하게 된다.

본 연구에서는 자원 상태 변화가 많은 그리드 환경의 특성을 고려하여 자원 상태의 변화 정도를 감시하여 모니터링 주기를 조절하고자 한다. 시스템 상황이 얼마나 자주 변하는지, 자원의 오버헤드 양에 따라, 자원 정보의 불확실성에 대한 손실률 등을 이용하여 주기를 계산하는 방법을 모델링 하였으며, 이를 이용해 주기를 실제 계산해서 적용해 보았다. 또한 시스템 사용률이 많고 적음을 모니터링 주기에 적용하는 것이 아니라 시스템 사용률의 변화 정도를 모니터링 주기에 적용하여 그리드 환경에 보다 적합한 모니터링을 제안한다.

2. 관련연구

현재 그리드 환경에서 그리드 모니터링 구조를 분석해 보면 그리드 타입의 요소들 사이에 모니터링에 필요한 데이터는 성능 문제의 소스를 결정하는 것 그리고 시스템과 어플리케이션을 변화시키는 것이다. 이 데이터들로부터 오류 발견과 회복 메커니즘을 결정된다.

현재 그리드 환경에서 범용적으로 사용되는 그리드 모니터링은 다음과 같은 구조로, 세 가지 구성요소를 보면 디렉토리 서비스는 정보를 발행하고 발견하도록 지원하고, 생산자(performance event source)는 성능 데이터를 이용하도록 만든다. 소비자 (performance event sink)는 성능 데이터를 받는다.[4]



(그림 1) 그리드 모니터링 컴포넌트

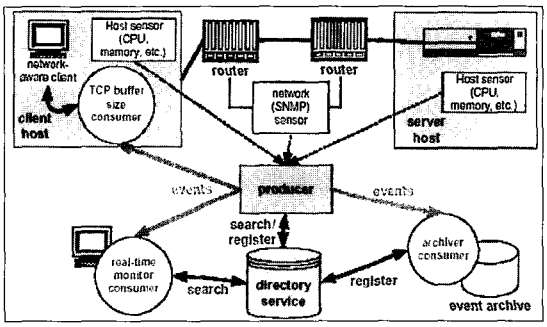
위의 그림의 컴포넌트 각각의 구조의 기능을 살펴보면, 생산자(producer)는 LDAP을 이용해 디렉터리 서비스에게 접근하여 자신의 존재를 알린다. 또한 소비자(consumer)는 관심 있는 생산자를 발견하기 위해 디렉터리 서비스(directory service)를 이용할 수 있다. 디렉터리 서비스는 생산자와 소비자의 위치를 알아내는데 이용된다.

각각의 컴포넌트의 기능과 인터페이스에 대해 더 정의해 보면 디렉터리 서비스의 기능은 대표적으로 알아보면 Add는 디렉터리에 엔트리 추가하는 것이며, Update는 디렉터리에서 엔트리 변화를 기록하는 것이고, Remove는 디렉터리로부터

엔트리 삭제한다. 또한 search는 어떤 선택된 규범을 근거로 해서 생산자 또는 소비자 찾도록 수행하는 기능을 갖고 있다.

생산자는 소비자에게 이벤트를 보내기 위해 사용하여 이벤트마다 다양한 생산자 인터페이스를 가질 수 있다. 소비자 는 생산자로부터 이벤트 데이터를 받기 위해 소비자 인터페이스를 이용한다.

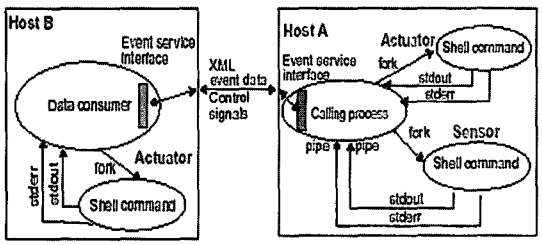
그리드 모니터링 구조는 time-stamped event와 같이 전송된 성능 데이터를 다루도록 설계됐으며, 이벤트는 이벤트 스키마에 의해 정의된 구조로 수집된 데이터의 타입이다.[5]



(그림 2) 모니터링 시스템의 구조

위의 그림은 시스템 모니터링이 사용된 예이다. 이벤트 데이터가 모든 네트워크 라우터 상에 있는 각각의 호스트 사이에서 수집되고, 생산자로 모은다. 그러면 생산자는 이용할 데이터를 디렉터리 서비스로 등록시킨다. A "network-aware" client는 잠재된 데이터를 이용해 TCP 버퍼 사이즈를 최적으로 한다.

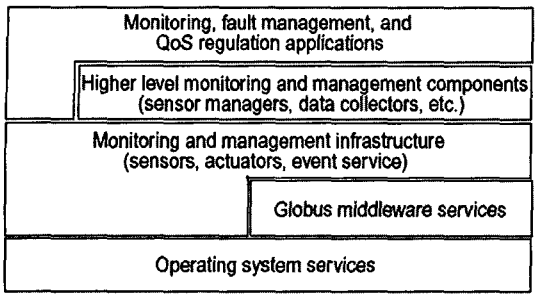
또 다른 그리드 환경의 모니터링으로 Information Power Grid [IPG]는 NASA에서 그리드 중 계산 그리드를 위한 좀더 효율적인 모니터링을 지원하고자 만든 인프라이다. 계산 그리드의 특성상 자원의 규모가 크고 복잡하기에 오류를 유발할 수 있는 가능성이 높은 특성을 고려하여 구축된 모니터링 시스템으로 기존의 그리드 모니터링보다 발전된 상태이다. 즉 그리드 환경에서 각자의 작업환경처럼 프로그램이 수행되어야 함으로써 잘못을 유발할 수 있는 조건을 찾도록 자원을 모니터링하는 메커니즘이 필요하고, 또한 QoS 조절하기 위한 회복 메커니즘도 요구되어 진다. 이러한 기능을 기반으로 디자인되어 있으며, 다음과 같이 구조로 이루어져 있다.[6]



(그림 3) 3가지 컴포넌트를 이용한 모니터링 시스템

세가지 기본적인 컴포넌트, 센서와 구동기, 이벤트 서비스로 구성되어 있다. 첫째 센서(sensor)는 타겟 시스템의 자원의 측정을 측정한다. 하나의 센서는 일반적으로 유닉스 유틸리티와 센서의 특정화된 측정을 실행한다. 이러한 측정은 센서마다 추정되는 수치로 표현된다. 내부센서는 리소스 사용정보를 불려진 프로세스 상에서 수집할 수 있다. 둘째 구동기(actuator)는 센서의 작업과 동일하지만, 다른 점은 특정화된 작업이나, 작업관리 또는 사용자 정의 작업을 수행하는데 shell 스크립트를 사용한다. 사용하는 구동기(actuator)는 kill process, send mail. shell command 수행, 일부의 글로버스 명령들을 포함한다. 셋째는 이벤트 서비스(event service)는 센서에서 수집된 정보를 이 정보에 관심이 있는 다른 작업공간에게 제공하는 메커니즘이다. 이벤트 서비스는 고객이 특정 정보와 그 정보를 위한 서버를 요구하기 위해 요청자와 제공자 범례를 갖고 있다. 또한 적용 특화된 이벤트 정보를 사용자 작업에서 소비자 작업으로 전달한다.

Information Power Grid [IPG]는 위의 기본적인 모니터링 구성 요소들이 API를 통해 요구하며, 이를 수행하는 소프트웨어의 구조는 다음 그림4와 같다.



(그림 4) Software architecture of monitoring and management infrastructure

3가지의 기본 컴포넌트는 통신과 보안을 위해 글로버스 미들웨어 서비스(Globus middle services)로 설계되어지고, high level의 모니터링은 3가지의 기본 컴포넌트를 기본으로 하여 회복 알고리즘과 오류 알고리즘이 설계된다.[7]

좀더 발전된 그리드 모니터링 시스템에서도 자원 정보를 좀더 정확하게 알기 위해서나 오류를 막기 위해서만 설계되어 있을 뿐, 모니터링이 가지고 있는 근본적인 문제였던 모니터링에 의한 자원의 오버헤드에 대해선 대책이 세워지지 못했다. 그리드 환경에서 자원을 사용자가 원하는 만큼 가장 정확한 정보를 제공하기 위해서는 항시 모니터링을 하는 것이 좋으나 실질적인 오버헤드가 너무 많아 효과적으로 모니터링하는 시스템이 필요하다.

본 연구에서는 그리드 환경에서 모니터링 주기가 자원 상태 변화 정도를 파악하고 자원의 상태 변화를 예측하지 못해 손실된 정도에 따라 모니터링 주기를 조절하여 최적의 주기로 모니터링하는 시스템을 제안하고자 한다. 이를 통해 모니터링에 의한 자원의 오버헤드를 줄일 수 있다.

3. 동적 모니터링 주기 (Dynamic Monitoring Interval)

단순히 컴퓨터만 피어로 참여하는 것이 아니라 대용량 저장 장치, 다양한 고성능 연구 장비들이 포함되어 있는 통합된 형태의 그리드는 네트워크 환경으로 연결된 컴퓨팅 자원을 사용자가 개인용 컴퓨터를 이용하듯이 사용할 수 있게 하기 위해 자원 정보 관리를 한다. 그리하여 자원 정보 관리 시스템에서는 자원의 정보를 모니터링하여 사용자가 원하는 만큼 자원을 할당해주고, 또한 사용자가 사용중인 자원에 대해서도 지속적으로 모니터링하여 이 자원에 문제가 발생하였을 경우를 대비하고, 문제가 발생했을 경우에는 모니터링된 자원 정보를 가지고 다시 재분배를 해 주는 것이 기본 메커니즘이다.[8]

자원 정보 관리 시스템에서 자원 정보를 발견하면 리스트에 넣고, 자원을 할당하려 할 때 검색하거나 모니터링을 하게 된다. 이렇게 모니터링된 자원은 하드웨어와 소프트웨어 정보, 동적인 시스템 상태 정보 등이 있으며, 이것을 자원 정보 관리 시스템에서 관리한다. 그리드 환경에서 자원 정보 관리 시스템에 의해서 자원이 할당되고 관리될 때 중요시 고려하는 요소는 CPU사용률, 메모리 사용률, 네트워크 상태 정보 등이다.

그리드 프로그램뿐만 아니라 분산 처리용 프로그램이 수행될 때, 자원 상태가 변화 정도가 CPU의 변화율에 종속적으로 나머지 모니터링 요소들이 변하므로 CPU 사용률의 변화를 살펴보면 자원의 상태 변화 정도를 예측할 수 있다. 즉 CPU의 사용률이 거의 변화가 없으면 다른 자원의 상태도 거의 변화가 없고 CPU사용률이 변화가 많다면 다른 자원 상태 또한 변화가 많다는 것이다. 다시 말하여 그리드 모니터링 이벤트들이 CPU의 변화율에 따라 종속적으로 나머지 모니터링 이벤트들이 변하므로 CPU 사용률이 많이 변했을 경우엔 나머지 모니터링 이벤트들도 많이 변하여 모니터링 시스템이 행해져야 한다. 또한 단순히 CPU의 사용률만을 보고 모니터링 시스템을 구동한다면 계속해서 많은 CPU를 사용하고 있는 시스템은 굉장히 짧은 주기로 모니터링이 이루어지게 될 것이며, 이는 자원에 오버헤드만 증가시킬 뿐이다. 그리하여 본 연구에서는 CPU 사용률의 변화가 거의 없을 경우에는 다른 모 또한 변화가 거의 없을 것이므로 모니터링할 필요가 없다는

것을 이용해 단지 CPU 사용률만으로 주기를 변경하는 것이 아니라 CPU 사용률의 변화율이 높고 낮음으로 모니터링의 여부를 결정하고 모니터링의 주기를 조절하자는 것이다.

CPU 사용률의 변화가 큰 경우에는 자원의 상태가 변화할 가능성이 크므로 모니터링을 하고, 모니터링 시스템의 모니터링 주기를 변경 조절하며, 자원 상태 변화 정보를 자원 정보 관리 시스템에게 알리는 것이다. 즉 CPU의 사용률이 높았다가 낮아지면, 자원 상태가 변한 것이므로 이 때 모니터링을 하고 또한 자원의 손실률이 작으므로 모니터링 주기를 길게 하는 것이다. 여기서 자원 정보의 상태변화율은 이전의 모니터링하여 알아온 CPU 사용률과 현재 모니터링된 CPU 사용률의 차를 통해 구하며, 이 차이가 큰 경우에는 시스템에 변화가 생겼다는 것으로 판단하고 모니터링 주기를 조절하도록 한다. [9]

CPU 사용률로 자원의 상태가 변화하였음을 알아보는 방법을 모델링 하면, 현재 모니터링을 통해 들어오는 CPU 상태 정보와 과거의 모니터링한 CPU 상태 정보를 비교하는 것이다. 현재 CPU가 사용하고 있는 CPU 사용률을 %로 모니터링해 오면, 다른 사용자에게 자원을 할당해 줄 수 있는 양은 전체 CPU 사용률 100%에서 현재 CPU 사용률을 뺀 값 e(k)가 된다. 즉 e(k-1)은 전단계에서 모니터링된 CPU 사용률을 뺀 %로 전단계에서 다른 사용자에게 할당할 수 있는 CPU 사용률이다. 또 본 절에서 설정한 변수는 현재의 자원 상태 변화뿐만이 아니라 과거의 history까지 고려한 PID(Proportional Integrated Differential) 컨트롤러를 이용한 dynamic threshold이다. 이는 PID 컨트롤러로 자원 정보 상태를 예측하는 u(k)를 정의한다.[10] 식 2는 PID 컨트롤러를 이용해 dynamic threshold로 u(k)를 표현한 것이다.

$$(k_p \times e(k) + k_i \times \sum_{j-1}^{k} e(j) + k_d \times (e(k) - e(k-1))$$

(식 1) PID 컨트롤러로 표현한 u(k)

이때 PID 컨트롤러는 각각 k_p =-0.5, k_i=0.125, k_d =-0.125로 조율을 맞춘다. 즉 u(k)를 PID 컨트롤러를 이용하여 현재 모니터링 된 CPU idle 값 e(k)로 나타낸 것이다. 또한 자원 상태 변화를 알기 위해 CPU 사용률의 변화량을 알아보면 이는 |u(k)-u(k-1)|를 통해 알 수 있다.

$$\begin{aligned}
&|u(k) - u(k-1)| \\
&= (k_p \times e(k) + k_i \times \sum_{j-1}^{k} e(j) + k_d \times (e(k) - e(k-1)) \\
&- (k_p \times e(k-1) + k_i \times \sum_{j=1}^{k-1} e(j) + k_d \times (e(k) - e(k-1)) \\
&= k_p \times (e(k) - e(k-1)) + k_i (\sum_{j=1}^{k} e(j) - \sum_{j=1}^{k-1} e(j)) \\
&\cong k_i \times e(k)
\end{aligned}$$

|u(k)-u(k-1)|의 값을 간단하게 정리해 보면, 이는 시스템의 현 상태를 PID컨트롤러로 나타낸 k_i*e(k) 결과 값과 비교해 볼 수 있다. 두 식을 비교해 볼 때 |u(k)-u(k-1)|이 k_i*e(k)보다 작거나 같으면 현재 CPU 사용률과 그 전의 CPU 사용률의 변화정도가 거의 없는 것이므로 CPU 사용률의 변화율이 거의 없는 안정된 상태를 의미한다. 반대로 |u(k)-u(k-1)|가 k_i*e(k)보다 크다면 현재 CPU 사용률과 그 전의 CPU 사용률의 변화정도가 큰 것으로 자원의 상태에 큰 변화가 있는 상태라는 것을 유추할 수 있다. 즉 CPU 사용률의 변화량을 PID컨트롤러를 통해 나타낸 |u(k)-u(k-1)|값과 k_i*e(k) 값을 비교하여 그리드의 자원 상태가 변화하였음을 파악함으로써 모니터링 시스템에 주기를 조절하고 모니터링이 이루어지도록 하는 것이다.

4. 최적 모니터링 주기 (Optimal Monitoring Interval)

그리드 환경에서는 자원관리 시스템이 모니터링 시스템을 통하여 자원 상태 정보를 계속해서 필요로 한다. 이렇게 자원 정보에 대한 모니터링

은 계속 일어나 매번 자원의 상태 정보를 업데이트한다. 실시간으로 자원의 상태 정보가 변하는 그리드 환경에서 자원에 대한 모니터링 시스템은 실시간으로 처리되는 것이 가장 정확한 정보를 갖겠지만, 방대한 자원의 정보를 실시간으로 처리하는 데 많은 자원이 사용됨에 따른 오버헤드 비용이 소모된다. 그러므로 모니터링 시스템이 얼마나 적합한 주기로 모니터링이 이루어지는 게 더 효율적인가에 대한 문제가 중요시 고려된다. 즉 실시간으로 모니터링을 한다면 가장 정확한 자원의 정보를 알 수 있겠지만, 자원의 성능에 대한 오버헤드가 클 것이며, 자원의 변화가 큰 상태에 시스템에서 모니터링 주기가 길다면 자원의 상태 정보를 정확하게 분석할 수가 없을 것이다. 또한 자원의 변화가 거의 없는 상태에서 모니터링 주기가 짧다면 자원 성능에 오버헤드가 될 뿐일 것이다. 그러므로 자원의 변화 상태 정도에 따라 능동적으로 모니터링 주기가 조절되는 것이 가장 최적의 모니터링 시스템일 것이다.

위의 절에서 자원의 상태변화를 파악하여 상태 변화율이 큰 경우에 모니터링 시스템이 이루어지도록 설계하였으며, 본 절에서는 자원의 상태 변화 그리드 환경에서 주기를 시스템 상황에 따라 변화를 주어 모니터링하여 정확한 자원의 정보를 제공하면서 모니터링의 비용, 오버헤드를 줄이자는 것이다. 그리드 환경에서의 가변적인 사항은 크게 세가지가 있다. 첫째 사용자가 원하는 자원을 제공하기 위해 자원 정보를 알아보는데 사용되는 모니터링 자원에 대한 오버헤드 비용, 둘째 실시간으로 자원 상태에 대한 정보를 갖고 있지 않기에 자원 정보를 잘못 파악하여 발생되는 손실률, 셋째 자원의 상황이 얼마나 가변적인지를 표시하는 자원 상태 변화율 등이 있다. 이런 3가지의 가변 사항을 고려하여 자원 상태에 따라 변화하는 모니터링 주기를 모델링 해 보자.

우선 3가지의 가변사항에 대한 변수 정의를 내려보면, 첫째 모니터링 주기 동안 자원의 상태가 얼마나 변화할 지에 대한 변화가능성을 p라 하자. 모니터링 주기동안 변화가 일어날 가능성 p는 시스템 상황에 따라 달라지게 되는데 시스템 상황은 사용자수, 프로세스의 수 등에 의해 좌우된다. 초기 시스템 상황을 파악하려면 실시간 모니터링을 하여 모니터링 주기 동안의 변화가능성 p를 샘플링 해야 하는데, 이는 실시간 모니터링과 모니터링 주기를 갖은 모니터링 시스템을 각각 실행시켜야 하는 오버헤드가 크다. 여기서는 변화가능성을 관리자가 현재 사용자 수와 프로세스 수 등을 고려하여 변화가능성을 초기화하는 것으로 한다.

두 번째는 주기적으로 모니터링을 하는 것 자체에 대한 시스템 자원 사용하는 오버헤드, M이다. 모니터링 시스템이 사용하는 자체 오버헤드 비용은 일정한 오버헤드 비용이기에 CPU, memory, net의 사용률의 합계를 %로 나타내 이용한다.

세 번째는 모니터링 주기동안 자원 상태 정보가 변화하여 제대로 작업을 수행하지 못해 발생한 손실률, 이를 L로 나타낸다. 즉 모니터링 주기 동안 20%의 변화가능성 p를 갖은 상황에서 자원 상태 변화가 40%였다면, 20%의 자원 상태 변화를 제외한 20% 자원 상태 변화는 작업을 제대로 수행하지 못해 발생한 손실률 L이 될 것이다.

네 번째는 최적의 모니터링 주기를 가리키는 t로서 이를 모델링 할 것이다.

본 논문의 제안된 4가지의 변수들 중에, 자원 상태 변화 가능성을 예측하는 p 값은 사용자에 의해 초기 값을 주고, 모니터링 시스템이 실행되면서 모니터링 시스템 자체가 갖는 오버헤드 비용 또한 시스템마다 주어진 값이므로, 시스템의 변화로 자원 상태를 예측하지 못하여 발생되는 손실량 L에 의해서 모니터링 주기t가 변하게 된다. 즉 사용자가 자원을 얻고자 한다면, 자원의 정보를 알기 위해 모니터링 자체가 사용하는 오버헤드 M, 자원 상태 정보가 얼마나 많이 변화되었는가를 나타내는 변화 가능성 p, 모니터링된 자원의 상태가 갑자기 변화하여 변화가능성 p보다 크게 되어 발생되는 손실률 L로 모니터링 주기 t를 조절하는 것이다.

그리드 환경에서 자원 상태 정보가 변화할 가능성 p와 모니터링 시스템 실행에 따른 모니터링 비용, 자원 상태가 변함을 예측하지 못해 발생한 손실률 L을 이용하여 다음과 같이 모니터링 주기에 대한 모델링을 해 보았다.

첫 단계로, 모니터링 주기를 t이라 하고 그 동안에 변화될 가능성을 t시간 동안 p의 확률로 나타낸다. 모니터링 주기 동안에 시스템이 변화할 가능성을 다음과 같이 나타낼 수 있다.

$$\sum_{i=0}^{t} pi = p\sum_{i=0}^{t} i = \frac{pt(t+1)}{2}$$

두 번째 단계로. 위의 모니터링 주기동안 자원의 상태 정보가 변화하여 발생하게 되는 손실 비용은 t시간 동안 변화될 가능성에 의한 손실량 L이므로 다음과 같이 나타낼 수 있다.

$$L\frac{pt(t+1)}{2}$$

세 번째 단계에서는 모니터링 주기 동안 모니터링이 실행되면서 사용한 자원 자체에 대한 오버헤드 비용을 포함하면, 모니터링 시스템이 구동될 때 모니터링 주기 동안에 발생한 모든 비용을 나타낸다. 즉 모니터링 주기동안의 총 비용인 것이며, 이는 다음과 같다.

$$L\frac{pt(t+1)}{2}$$

네 번째 단계로 위의 모니터링 주기 동안의 모든 비용을 단위 시간당 사용한 비용으로 나타낸 것으로, 총비용을 t시간으로 나눈 게 된다. 즉, 시간당 모니터링의 비용이 다음과 같다.

$$c = \frac{M + L\frac{pt(t+1)}{2}}{t}$$

다섯 번째 단계로 최적의 모니터링 주기 t의 값을 구하기 위해 단위 시간당 모니터링에 사용된 비용 C의 값을 최소화 해야 한다. 즉 미분을 통해 모니터링 비용을 최소화하여 식을 간단히 정리하면 식[1]과 같이 나온다.

$$\frac{\partial C}{\partial t} = \frac{Lp}{2} - \frac{M}{t^2} = 0 \; t_{opt} = \sqrt{\frac{2M}{Lp}}$$

(식 2) 최적의 주기 topt

식 1의 계산 결과처럼 topt 을 간단하게 계산하여 손실률에 따른 모니터링 주기 조절을 할 수 있다. 즉, 자원 상태의 변화율이 작아서 자원의 손실률 L이 작아지면 모니터링 주기 topt 는 길어진다.

5. 실험 및 결과

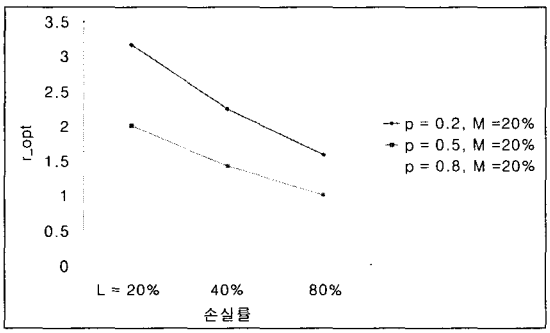
3절, 4절을 통해 모델링한 이론을 바탕으로 실험을 해보았다. 크게 3가지의 실험을 해보았는데 첫 번째는 최적의 모니터링 주기에 대한 실험이며, 두 번째는 CPU의 상태 변화율을 측정해 cpu 상태 변화에 따라 자원 상태의 변화를 의미함으로 모니터링을 해야 하는 시점을 알려주는 실험이다. 세 번째는 위의 2가지 제안을 적용한 모니터링 시스템과 그리드 환경에서 범용적으로 사용하는 일정 주기 모니터링을 비교하는 실험이다.

첫 번째 실험은 최소한의 모니터링 비용을 구하기 위해서 위에서 제안한 가변적인 사항을 적용해 적합한 topt을 찾아보자. 보내려는 자원을 멀티미디어 데이터라고 할 때 멀티미디어 데이터의 전송 시 잃어버린 패킷 수를 비율로 바꾸어 손실률 L로 하고, 모니터링 시스템에 의해서 사용된 자원의 overhead, 비용을 CPU, 메모리와 네트워크의 사용률의 평균 값으로 구하여 모니터링 비용 M이라 하자. 또한 자원 정보의 변화 가능성을 p라 놓고, 이 p 값에 따라, 손실률 L에 따라, 모니터링에 의한 overhead M에 따라 최적의 t을 계산해 보자.

(표 1) optimal 모니터링

L= 5일경우-20%			L= 10일경우-40%			L= 20일경우-80%		
P	M(%)	t_{opt}	p	M(%)	t_{opt}	p	M(%)	t_{opt}
0.2	20	3.1623	0.2	20	2.2361	0.2	20	1.5811
0.2	50	5	0.2	50	3.5355	0.2	50	2.5
0.2	80	6.3246	0.2	80	4.4721	0.2	80	3.1623
0.5	20	2	0.5	20	1.4142	0.5	20	1
0.5	50	3.1623	0.5	50	2.2361	0.5	50	1.5811
0.5	80	4	0.5	80	2.8284	0.5	80	2
0.8	20	1.5811	0.8	20	1.118	0.8	20	0.7906
0.8	50	2.5	0.8	50	1.7678	0.8	50	1.25
0.8	80	3.1623	0.8	80	2.2361	0.8	80	1.5811

실험 결과를 살펴보면, 손실률 L와 이때 모니터링에 의한 오버헤드 M가 같은 경우 자원의 정보가 변화할 가능성이 커질수록 변화가 잦은 시스템이므로 좀더 짧은 주기를 주어야 정확한 자원 정보를 얻을 수 있을 것이다. 또한 같은 자원의 정보가 변할 가능성을 가질 때 손실률 L가 클수록 주기 짧아져야 정확한 자원 정보를 얻을 수 있을 것이다. 이를 알아보기 쉽도록 모니터링에 의해 overhead M가 20%일 때 자원의 변화 가능성이 각각 0.2, 0.5, 0.8일 때 자원의 정보가 변화하여 손실되는 정도 L를 차트로 표현해 보면 다음과 같다.



(그림 5) 모니터링에 대한 오버헤드가 20%인 경우

차트를 보면 자원의 성능 변화 상태 p가 0.2에서 0.8로 증가하면, 자원의 성능 상태 변화 p가 크다는 것이므로 자원의 모니터링 주기 t_{opt}는 짧아지고 있다. 또한 같은 자원의 성능 변화 p를 가질 때, 즉 자원의 성능변화 p가 0.5라 한다면 이때 자원 정보의 변화로 인한 손실률 L이 20%에서 80%이 증가함에 따라 모니터링 주기 t_{opt}도 짧아진다.

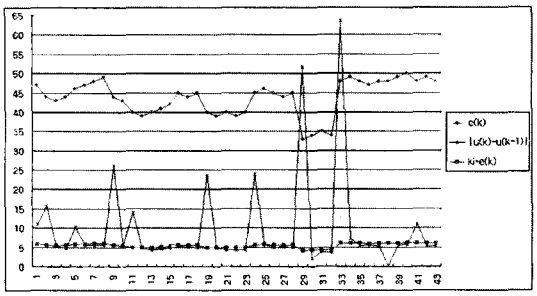
두 번째로 CPU 상태를 파악하여 자원 정보 상태가 변했으므로 모니터링이 이루어져야 함을 증명하는 실험으로 CPU의 상태 변화가 많은 경우와 많이 않은 경우에 각각 적용시켜 보았다.

첫번째 적용 환경으로 자원 정보 변화 상태가 크지 않는 smooth한 경우이다. 이 경우에는 자원 정보 변화가 작더라도 변한 상태이므로 이를 모니터링 주기에 반영하여 모니터링 주기를 조절할 필요가 있다.

(표 2) 시스템 변화율이 smooth한 경우

smooth monitor					
cpu사용률	e(k)	u(k)	u(k-1)	|u(k)-u(k-1)|	k_i*e(k)
53	47	117.75	128.75	11	5.875
57	43	96.875	102.125	5.25	5.375
56	44	101.625	96.875	4.75	5.5
54	46	111.875	101.625	10.25	5.75
53	47	117.375	111.875	5.5	5.875
52	48	122.875	117.375	5.5	6
51	49	128.5	122.875	5.625	6.125
61	39	78.125	82.875	4.75	4.875
60	40	82.375	78.125	4.25	5
59	41	87	82.375	4.625	5.125
58	42	91.75	87	4.75	5.25
53	47	116.875	91.75	25.125	5.875
54	46	112.25	116.875	4.625	5.75
55	45	107	112.25	5.25	5.625
56	44	101.875	107	5.125	5.5
55	45	106.75	101.875	4.875	5.625
60	40	83.125	106.75	23.625	5
61	39	78.125	83.125	5	4.875
60	40	82.375	78.125	4.25	5
61	39	78.125	82.375	4.25	4.875
60	40	82.375	78.125	4.25	5
55	45	106.25	82.375	23.875	5.625
54	46	112	106.25	5.75	5.75
55	45	107	112	5	5.625
56	44	101.875	107	5.125	5.5
55	45	106.75	101.875	4.875	5.625
66	34	57.25	55.125	2.125	4.25
65	35	61.125	57.25	3.875	4.375
66	34	57.5	61.125	3.625	4.25

위의 표를 분석해보면, 자원 정보 변화 정도가 1에서 5정도로 작은 경우, 1,2정도의 변화보다 4,5정도의 변화가 크므로 그 변화를 알아야 하는데, 이때 사용 가능한 자원 정보 변화가 갑자기 40에서 45로 늘어난 경우 |u(k)-u(k-1)|의 값이 k_i*e(k)보다 크므로 자원 정보 상태 변화가 있는 것이므로 모니터링 주기를 줄이도록 조절한다. 위의 표를 간단하게 차트화를 시키면, |u(k)-u(k-1)|값이 갑자기 변화한 경우 k_i*e(k)의 그래프도 변화가 크게 나타난다는 것을 알 수 있다.



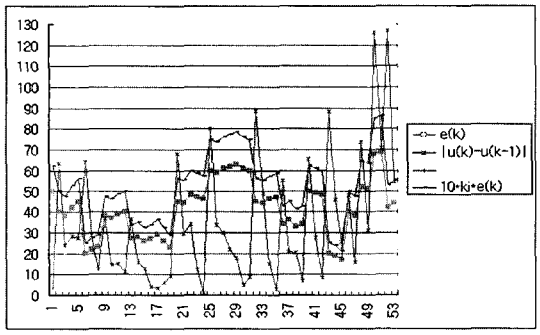
(그림 6) smooth한 시스템 변화에서의 변화가능성

두 번째 적용 환경은 자원 정보 상태가 굉장히 많이 변화하는 경우이다. 예를 들어 자원의 정보가 계속해서 크게 5에서 20정도로 변화 할 경우, 앞의 환경보다 변화가 크다고 모니터링 주기를 짧게 하여 체크하면 overhead가 크게 된다. 이때는 지금 현 상황에서 CPU 사용률의 작은 변화와 큰 변화를 파악하여 자원 정보 상태가 변화했음을 체크해야 한다.

(표 3) 시스템 변화율이 dynamic한 경우

monitor					
cpu사용률	e(k)	u(k)	u(k-1)	|u(k)-u(k-	ki*e(k)
50	50	-2.5	1.25	3.75	6.25
60	40	83.75	-2.5	86.25	5
62	38	73.875	83.75	9.875	4.75
58	42	91.375	73.875	17.5	5.25
55	45	106.5	91.375	15.125	5.625
80	20	19.375	106.5	87.125	2.5
78	22	20.375	19.375	1	2.75
77	23	22.875	20.375	2.5	2.875
62	38	71.75	22.875	48.875	4.75
63	37	69.5	71.75	2.25	4.625
61	39	77.75	69.5	8.25	4.875
60	40	82.375	77.75	4.625	5
73	27	35.375	82.375	47	3.375
72	28	36.625	35.375	1.25	3.5
74	26	31.125	36.625	5.5	3.25
73	27	33.625	31.125	2.5	3.375
71	29	39.625	33.625	6	3.625
74	26	31.25	39.625	8.375	3.25
77	23	23.375	31.25	7.875	2.875
55	45	104.125	23.375	80.75	5.625
56	44	101.875	104.125	2.25	5.5
52	48	122.5	101.875	20.625	6
53	47	117.625	122.5	4.875	5.875
54	46	112.25	117.625	5.375	5.75
40	60	197	112.25	84.75	7.5
41	59	191.875	197	5.125	7.375
39	61	205.625	191.875	13.75	7.625
38	62	213	205.625	7.375	7.75
37	63	220.375	213	7.375	7.875
39	61	206.125	220.375	14.25	7.625

위의 표를 분석해보면, 자원 정보 변화 정도가 |u(k)-u(k-1)|의 값이 k_i*e(k)보다 작은 경우는 자원 상태 변화가 작으므로 모니터링 주기를 길게 하고, 큰 경우에는 자원 상태 변화가 크므로 모니터링 주기를 짧게 하는 것이다. 예를 들어 자원 정보 사용 가능한 정도가 46%에서 60%로 증가하면 자원 정보 상태가 변한 것으로 |u(k)-u(k-1)|의 값이 k_i*e(k) 보다 크게 된다. 위 표를 차트화 하면 다음과 같다.

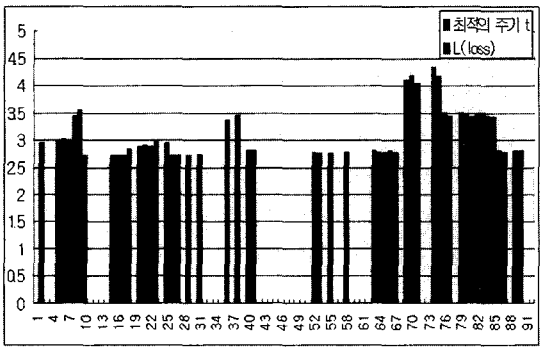


(그림 7) dynamic한 시스템 변화에서의 변화가능성

|u(k)-u(k-1)|값이 변화가 굉장히 큰 경우에서도 더 큰 변화에 뾰족하게 나타나는 순간이 있다. 이때 |u(k)-u(k-1)|값이 k_i*e(k)보다 크게 되어 자원 정보 상태의 변화를 파악하여 모니터링 주기를 짧아지도록 적용시킨다. 특정 환경에 5% threshold로 변화를 detect했을 경우 100번의 모니터링을 수행했을 때, 본 연구를 통해 모니터링의 오버헤드를 20%정도 줄이는 효과를 볼 수 있다.

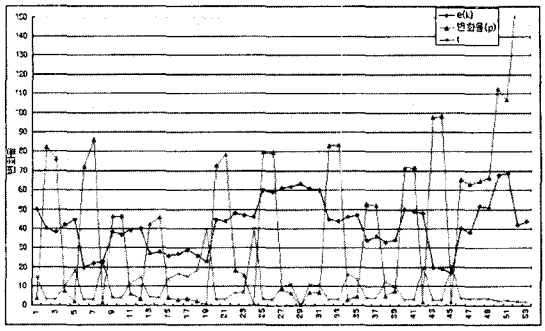
세 번째 실험은 자원의 상태 변화에 따라 본 논문에서 제안한 모니터링 시스템이 적합한지를 알아보기 위한 실험으로 그리드 환경에서 똑같은 사양의 클러스터에서 각각 1초마다 모니터링되는 시스템과 본 제안을 적용한 모니터링 시스템을 비교해 보았다. 본 실험에 사용된 클러스터는 펜티엄III CPU 820, 메모리 256 M, 하드 10GB이다. 1초 마다 모니터링되는 시스템을 통하여 현재 자원의 상태를 정확하게 파악할 수 있으므로, 본 논문에서 제안한 방식이 자원의 손실률에 따라 얼마나 적합하게 모니터링 주기를 변동시키는지 비교해 볼 수 있을 것이다.

첫번째 실험 그림에서는 1초마다 모니터링을 한 시스템과 비교하여 본 논문에서 제안한 방식의 모니터링 시스템에서의 자원에 손실률에 따른 최적의 주기의 변화율을 본 결과이다. 자원의 손실률이 클 경우에는 모니터링 주기를 작게 조절하여 다음의 모니터링 주기에 의한 손실률이 작아지게 하였으며, 본 논문에서 제안된 손실 예정량보다 실제 손실량이 없을 경우에는 모니터링 시스템이 체크하지 않아 모니터링에 의한 오버헤드를 줄이도록 하였다. 막대그래프가 나타나지 않은 것이 자원의 손실률로 인한 시스템의 변화율에 변화를 주지 않을 경우를 나타내며, 이는 모니터링 자체가 갖는 자원에 대한 오버헤드를 줄인 부분이 된다.



(그림 8) 손실률L에 따른 최적의 주기t

두 번째 실험 그림은 본 논문에서 제안한 모니터링 시스템으로 이렇게 자원의 손실률L에 따라 자원의 변화율 파악하여 1초마다 모니터링하여 얻은 실제의 자원 상태와 비교한 것이다. 실제의 자원 상태와, 상태 변화율에 따라 본 제안의 모니터링 주기의 변화가 같은 모양의 그래프를 이루면서 변화하는데, 이는 모니터링이 갖는 자체 모니터링의 오버헤드를 줄인 최적화된 모니터링 시스템으로 실제 그리드 자원상태를 정확하게 파악할 수 있음을 의미한다. 그러므로 일정한 주기를 갖는 모니터링 시스템보다 나은 모니터링 성능임을 알 수 있을 것이다.



(그림 9) 실제 자원의 상태 변화율p에 따른 주기 t의 변화

하지만 본 제안된 모니터링 시스템은 자원 상태의 변화율이 단시간내의 다이나믹한 그래프를 나타낼 때는 이 상태를 파악하지 못하는 단점이 있다. 그러나 이 모니터링 시스템을 사용할 그리드 환경은 자원 상태의 변화가 한번 자원의 할당을 받고 반환하는데 있어서 짧은 시간의 다이나믹한 변화 그래프를 그리지 않고, smooth한 변화 상태를 보이기 때문에 본 논문에서 제안한 모니터링 시스템을 적용하는데 문제가 없다.

5. 결 론

그리드 환경에서 자원 관리 시스템이 자원의 정보를 분석하고 관리하기 위해서는 반드시 모니터링 과정이 선행되어야 한다. 또한 효율적인 자원관리를 위해서도 자원 정보에 관한 모니터링은 반드시 이루어져야 한다. 여기서 문제는 반드시 이루어져야 하는 자원 정보에 대한 모니터링을 얼마나 자주 하느냐 하는 것이다.

현재 그리드의 모니터링 시스템은 그리드 환경의 유동성이 크고, 실시간으로 달라질 수 있는 가능성을 무시한 채 일정한 주기를 주어 자원 정보에 대한 모니터링을 하고 있다. 다시 말해서 그리드 환경의 자원 성능 상태와는 상관없는 모니터링을 하는 것이다.

본 연구는 자원 정보의 상태 변화율을 파악하기 위해 CPU 상태를 관찰하고 이의 변화 정도를

이용하여 자원 상태를 유추한다. 이렇게 자원의 상태가 변화하였음을 발견하면, 이때 자원의 대한 모니터링을 실시하고 모니터링 주기를 자원의 상태 변화율과 모니터링의 잘못된 예측으로 인한 손실률을 이용해 조절하였다. 또한 본 논문에서 제안된 모니터링 모델링을 현재 그리드 모니터링 기반을 두고, 새로운 하나의 모듈을 추가함으로써 모니터링의 주기를 자원 상태 변화율에 따른 optimal 모니터링 주기를 갖도록 한다. 즉, 본 논문을 적용하는 면에 있어서도 현재의 구성요소와 코드에 변화 없이 적용하기에 쉽게 최적화된 모니터링 주기로 자원 관리를 할 수 있다.

참고문헌

[1] I. Foster., "The Grid: A New Infrastructure for 21st Century Science.", Physics Today, 55(2):42-47 2002.

[2] A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunst, M. Ripeanu, B, Schwartzkopf, H, Stockinger, K. Stockinger, B. Tierney, "Giggle: A Framework for Constructing Scalable Replica Location Services.", Proceedings of Supercomputing 2002 (SC2002), November 2002.

[3] Tierney, B., B. Crowley, D. Gunter, M. Holding, J. Lee, M. Thompson "A Monitoring Sensor Management System for Grid Environments" Proceedings of the IEEE High Performance Distributed Computing conference (HPDC-9), August 2000, LBNL-45260.

[4] B. Tierney, R. Aydt, D. Gunter, W. Smith, M. Swany, V. Taylor, R. Wolski, "A Grid Monitoring Architecture", Global Grid Forum Performance Working Group, March 2000

[5] TopoMon: A Monitoring Tool for Grid Network Topology, Proceedings of ICCS 02.

[6] A. Waheed, W. Smith, J. George, J. Yan., "An Infrastructure for Monitoring and Management in Computational Grids", In Proceedings of the 2000 Conference on Languages, Compilers, and Runtime Systems.

[7] Ashvin Goel, David Steere, Calton Pu, and Jonathan Walpole, "Adaptive Resource Management Via Modular Feedback Control", Submitted to HOTOS 1999

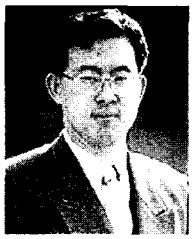
[8] A. Waheed, W. Smith, J. George, J. Yan., "An Infrastructure for Monitoring and Management in Computational Grids", In Proceedings of the 2000 Conference on Languages, Compilers, and Runtime Systems.

[9] monitoring event, "Discovery and Monitoring Event Description(DAMED-WG)", http://www-didc.lbl.gov/ damed/

[10] PID Tutorial, http://www.engin.umich.edu/group/ctm/PID/PID.html

◐ 저 자 소 개 ◑

김 혜 주
2002년 서울여자대학교 컴퓨터공학과 졸업(학사)
2004년 서울여자대학교 대학원 컴퓨터공학과 졸업예정(석사)
관심분야 : 네트워크, 그리드, 임베디드 시스템, 요구공학 etc.
E-mail : judy012@swu.ac.kr



허 의 남
1990년 부산대학교 전산통계학과 졸업 (학사)
1995년 U. of Texas at Arlington 컴퓨터공학과 졸업(석사)
2002년 Ohio University 대학원 컴퓨터학과 졸업(박사)
2003년~현재 : 서울여자대학교 정보통신공학부 교수
관심분야 : 그리드, 네트워크, QoS, 실시간 시스템, 정보보안 etc.
E-mail : huh@swu.ac.kr

이 웅 재
1981년 연세대학교 전산학과 졸업(학사)
1985년 Univ. of Illinois at Chicago전산학과 졸업(석사)
1991년 Illinois Institute of Technology 졸업(박사)
1993년~현재 : 서울여자대학교 정보통신공학부 교수
관심분야 : 인공지능, 자연어처리, 멀티미디어 etc.
E-mail : wjlee@swu.ac.kr

박 형 우
1985년 서울대학교 졸업(학사)
1996년 성균관대학교 대학원 졸업(석사)
2001년 성균관대학교 대학원 졸업(박사)
2001년~현재 : 한국과학기술정보연구원 그리드실장
관심분야 : 그리드, 차세대 TCP etc.
E-mail : hyoark@kisti.re.kr