# CC-NUMA 시스템을 위한 진단 소프트웨어 개발

## (Development of Diagnosis Software for CC-NUMA Systems)

정 태 일 † 정 낙 주 †† 김 주 만 ††† 김 해 진 ††††
(Taeil Jeong) (Nahkju Jeong) (Jooman Kim) (Haejin Kim)

**요 약** 본 논문에서는 CC-NUMA 시스템을 위한 진단 소프트웨어에 대한 구현 방법 및 결과를 소개하였다. CC-NUMA 구조는 두 대 이상의 SMP 시스템들에 캐쉬 일관성을 유지하기 위한 하드웨어를 장착하고, 이들을 고속 연결망으로 연결함으로써 시스템의 성능 및 확장성을 향상시켜 준다. 그러나, CC-NUMA 시스템은 운영체제 측면에서는 단일 시스템 이미지로서 보여지는 반면, 하드웨어 구조와 밀접한 진단 소프트웨어에서는 이를 별개의 시스템으로 고려해야 한다. 이와 같은 구조 때문에 기존의 상용 관리 소프트웨어로는 CC-NUMA 시스템에 대한 진단 및 관리를 하기 어렵다. 또한, TCO(Total Cost of Ownership) 절감 측면에서 최근 대두되고 있는 원격 진단 및 관리의 필요성이 증가하고 있다. 본 논문에서는 이러한 요구사항에 따라 CC-NUMA 구조에 적합한 진단 소프트웨어 모듈을 설계하였으며, 원격 진단 및 관리가 용이한 클라이언트-서버 구조의 진단 메커니즘을 제시하였다. 또한, 관리자가 어느 시스템에서도 서버에 접근할 수 있는 접근성을 향상시키기 위하여 자바 기반 사용자 인터페이스를 채택하였다.

***Abstract*** This paper introduces an implementation of the diagnosis software for CC-NUMA systems. The CC-NUMA architecture is composed of two or more SMP nodes installed with the specialized hardware to provide cache-coherent operation and the high-speed interconnection network to connect each node, it enables both the high performance and the high scalability. While the CC-NUMA system provides the single system image in the operating system aspect, it should be considered the multiple systems by the diagnostic software. Thus it is difficult to diagnose and manage CC-NUMA system using commercial administration software due to characteristics of the complicated architecture. The remote diagnosis and management are also required with a view to reduce Total Cost of Ownership. In this paper, we design diagnostic software to manage CC-NUMA server system, and propose its mechanism in client-server manner to support remote administration. Additionally, we use the Java-based user interface to enlarge an administrator's accessibility.

## 1. 서 론

대규모 비즈니스 컴퓨팅 환경에서는 수많은 컴퓨터, 데이타 및 인력이 하나의 네트워크로 연결되어 있기 때문에 이들에 대한 효율적 관리가 중요한 이슈로 대두되고 있다. 예를 들어, 어느 한 시스템이 사용 불가능해졌을 때, 이로 인한 직접적인 피해 뿐만 아니라 이를 해결하기 위한 노력에 대한 비용까지도 전체적인 손실 비용에 포함시키고 있다. 따라서, 이러한 사태를 사전에 방지하고, 문제가 발생했을 때 이를 최소한의 비용으로 처리하고자 하는 요구가 발생하고 있다. 이와 같이, 유기적인 조직 체계에서 조직원 및 조직에 포함된 모든 자원을 운용하는 총 비용을 TCO(Total Cost of Ownership)라 하며, 이를 줄이기 위한 노력이 다양하게

† 정 회 원 : 한국전자통신연구원 운영체제연구팀 연구원
taeil@etri.re.kr
†† 비 회 원 : 한국전자통신연구원 리눅스연구팀 연구원
njjeong@etri.re.kr
††† 비 회 원 : 한국전자통신연구원 운영체제연구팀 연구원
jmk@etri.re.kr
†††† 종신회원 : 한국전자통신연구원 리눅스연구팀 연구원
hjkim@computer.etri.re.kr
논문접수 : 1999년 6월 7일
심사완료 : 1999년 10월 19일

전개되고 있다. 특히, 컴퓨팅 환경에서 하드웨어와 소프트웨어에 대한 유지 보수 비용을 줄이기 위한 방안이 하드웨어 업체 및 소프트웨어 업체들에 의해 제시되고 있다.

이러한 방안으로는 주로 시스템 유지 보수의 자동화 및 표준화 등이 주류를 이루고 있다.

인텔(Intel)에 의해 주도되고 있는 WfM(Wired for Management)[1]은 시스템 자체적으로 관리 기능을 제공하기 위한 방안으로서 제시된 사양이며, DMTF (Desktop Management Task Force)에 의해 제시된 DMI(Desktop Management Interface)[2]는 시스템 관리 대상을 정의하고 관리 소프트웨어가 하드웨어 업체와 관계없이 하드웨어를 관리할 수 있도록 인터페이스를 정의해 놓은 표준이다. 또한, IPMI(Intelligent Platform Management Interface)[3]는 인텔, HP, NEC, 델(Dell) 등에 의해 시스템 내부에서 시스템 관리 마이크로 컨트롤러 간의 인터페이스를 정의한 것이다. 또한, 웹(web) 인터페이스를 통한 표준화를 지향하는 WBEM(Web Based Enterprise Management)[4] 등도 제시되었다.

결국, 중형 서버급 시스템을 개발함에 있어서 시스템 관리 기능은 부수적인 기능이 아니라 필수적인 기능이어야 하며, 이를 지원하는 하드웨어 및 소프트웨어의 개발은 필수적이다.

실제로 SMP(Symmetric Multi-Processor) 구조의 서버를 위한 시스템 진단 및 관리 소프트웨어는 많이 개발되어 있다. 그러나, 최근 확장성 및 고성능을 목적으로 제시되고 있는 CC-NUMA(Cache Coherent-Non Uniform Memory Access) 서버에 대해서는 상이한 구조적 특성 때문에 기존의 SMP 용 소프트웨어를 사용할 수 없으며, CC-NUMA 서버를 위한 진단 및 관리 소프트웨어라 할 지라도 자체 모델에 특화되어 있기 때문에 일반적인 시스템에서는 사용하기 어렵다.

따라서, 본 논문에서는 현재 개발 중인 CC-NUMA 서버에 대하여 하드웨어 종속성을 탈피하고 부가적으로 원격 진단 및 관리 기능을 제공하기 위한 진단 메커니즘을 설계하고 구현하고자 한다.

본 논문의 구성은 다음과 같다. 2 장에서는 CC-NUMA 시스템 및 이를 진단 및 관리하기 위한 진단 소프트웨어들을 소개하고, 3 장에서는 본 논문에서 제시된 CC-NUMA 시스템에 적합한 진단 메커니즘의 설계 내용을 다룬다. 또한, 4 장에서는 설계된 진단 메커니즘의 구현 내용에 대해 설명하고, 5 장에서 결론을 맺는다.

## 2. CC-NUMA 시스템에서의 진단

### 2.1 CC-NUMA 시스템

기존의 고성능 서버는 주로 SMP(Symmetric Multi-Processor) 구조와 MPP(Massively Parallel Processor) 구조로 구현되어 왔다. 그러나, 최근 NUMA(Non-Uniform Memory Access) 기술[7]이 보편화되면서 SMP와 MPP의 장점을 복합한 CC-NUMA 형 서버가 개발되고 있다. 이러한 CC-NUMA 시스템은 두 개 이상의 SMP 노드와 이를 연결하기 위한 고속 연결망으로 구성되어 있으며, 각 노드에는 캐쉬 일관성을 지원하기 위한 하드웨어를 장착하고 있다. 그림 1은 CC-NUMA 시스템의 일반적인 구조를 보여준다.

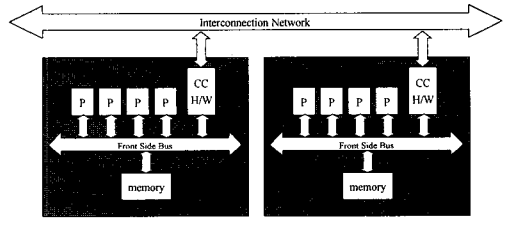



그림 1 CC-NUMA 시스템 구조

그림 1에서 보이는 바와 같이, 기존의 SMP 시스템과 마찬가지로 각 노드 내의 모든 프로세서들은 FSB(Front Side Bus)를 통해 메모리를 공유한다. NUMA 구조에서는 노드 내의 지역 메모리(local memory)뿐만 아니라, 상호 연결망(interconnection network)을 통해 다른 노드의 원격 메모리(remote memory)도 액세스할 수 있도록 별도의 하드웨어가 각 노드의 FSB 상에 장착된다. 특히, CC-NUMA 시스템에서는 캐쉬 일관성을 위한 하드웨어가 각 노드에 장착되어야 하며, 그림 1에서는 CC H/W로 표현되어 있다. 이러한 하드웨어를 이용하여 CC-NUMA 시스템에서는 모든 노드의 메모리를 공유함으로써 운영체제 레벨에서 단일 시스템 이미지(single system image)를 제공한다.

결국, CC-NUMA 시스템은 SMP 시스템의 확장성을 크게 개선시키면서도, 기존 SMP 애플리케이션을 그대로 사용할 수 있다는 장점을 가질 수 있다.

### 2.2 SMP 시스템을 위한 진단 및 관리 소프트웨어

중형 서버급 시스템에서는 시스템의 성능 뿐만 아니라 서버의 가용성 및 안전성이 매우 중요시된다. 즉, 서버의 다운 타임(down time)을 최소화하고, 이를 유지 보수하는 기간 및 비용을 줄이는 것이 서버 개발자 및

운영자의 주요 관심사가 되고 있다.

서버의 진단 및 관리 소프트웨어는 관리 방법 및 범위에 따라 크게 하드웨어를 통한 진단, 운영체제를 통한 시스템 관리, 분산 환경에서의 네트워크 관리 등으로 구분할 수 있다.

먼저, 하드웨어를 통한 진단은 시스템에 진단을 위한 하드웨어를 포함시킨 후, 이러한 하드웨어를 통해 시스템의 상태를 모니터링(monitoring) 또는 제어하는 방법이다. 이러한 방법은 시스템 제작자들에 의해 추진되고 있으며, 인텔 등에 의해 추진되고 있는 IPMI(Intelligent Platform Management Interface)가 대표적이다. IPMI에서는 서버 보드의 진단 컨트롤러(diagnostic controller) 간의 인터페이스 및 진단 컨트롤러와 호스트 간의 인터페이스를 정의하고 있다. 따라서, IPMI를 지원하는 서버 시스템에 대해서는 IPMI 인터페이스에 따라 시스템을 제어하기 위한 소프트웨어가 필요하다. 이러한 소프트웨어로는 인텔의 ISC(Intel Server Control)[8]가 있다. 그러나, 이러한 종류의 소프트웨어들은 하드웨어에 매우 밀접하게 종속되어 있기 때문에 범용성이 떨어지며, 원격 제어 등의 기능을 제공하지 않기 때문에 주로 서버 시스템의 콘솔(console)을 통해 동작되어야 한다.

운영체제를 통한 시스템 관리는 운영체제 차원에서의 자원을 관리하는 것이다. 이러한 자원에는 주로 사용자, 디스크 등의 저장장치, 프로세스 관리 등이 주로 포함되지만, 바이오스(BIOS) 및 운영체제의 시스템 콜(system call) 등을 통해 시스템 다운(system down), 리부팅(rebooting) 등의 기능을 제공해준다. 이러한 관리 소프트웨어의 예로는 SCO UnixWare 7 상의 SCOAdmin[9] 및 마이크로소프트(Microsoft) Windows NT 상의 관리 마법사 등이 있다. 그러나, 이러한 시스템 관리는 운영체제에 의존적이므로 주로 운영체제에서 제공되는 관리 기능이 포함되어 있으며, 하드웨어와 관련된 시스템 진단 기능은 매우 제한적이다.

분산 시스템 환경에서 여러 대의 시스템이 네트워크로 연결되어 있을 때, 이들을 종합적으로 관리하기 위한 소프트웨어로서 네트워크 관리 소프트웨어들이 있다. 이러한 소프트웨어의 예로는 CA(Computer Associates)의 Unicenter-TNG[10], HP의 NNM(Network Node Manager)[11] 및 하드웨어 솔루션을 포함한 인텔의 LANDesk[12] 등이 있다. 이러한 소프트웨어들은 각 시스템을 독립된 별개의 시스템으로 인식하므로 운영체제 상위 레벨에서 시스템 관리를 수행한다.

### 2.3 CC-NUMA 시스템을 위한 진단 소프트웨어

2.1 절에서 언급한 바와 같이 CC-NUMA 시스템은 두 개 이상의 SMP 노드가 메모리를 공유함으로써 단일 시스템 이미지를 제공한다. 따라서, 하나의 CC-NUMA 시스템에는 하나의 운영체제가 동작하므로, 이러한 운영체제 상의 모든 애플리케이션은 어떤 노드에서 동작하게 될 지 전혀 예측할 수 없다.

또한, 2.2 절에서 나열한 일반적인 진단 및 관리 소프트웨어들은 CC-NUMA 시스템과 같은 시스템 구조를 지원하지 않고 있다. 즉, CC-NUMA 구조는 운영체제 측면에서는 단일 노드이지만 시스템 구조 상으로는 다중 노드들로 구성되어 있기 때문에 이러한 소프트웨어로는 하나의 노드만을 진단할 수 밖에 없다.

결국 CC-NUMA 시스템을 제작하는 주요 공급자들은 기존의 관리 소프트웨어에 의존하지 않고, 자체 시스템을 위한 진단 및 관리 소프트웨어를 개발하고 있다. 대표적인 예로는 DG(Data General)의 AV20000 시스템을 위한 M3D(Monitor, Manage, and Maintain) 소프트웨어[13]와 시퀀트(Sequent)의 NUMA-Q 시스템을 위한 VCS(Virtual Console Software)[14] 등이 있다. 이러한 소프트웨어들은 진단 및 관리 기능 뿐만 아니라 CC-NUMA 초기화 기능 등의 부가적인 기능을 포함하고 있다.

그러나, 이러한 소프트웨어들은 자체 시스템 구성에 적합하도록 설계되었을 뿐만 아니라, DG/UX 및 DYNIX/ptx 등 자체 운영체제에서만 구동된다. 따라서, 이러한 소프트웨어들의 이식성(portability)은 매우 낮다고 할 수 있다.

본 논문에서는 범용성을 가지는 SHV(Standard High Volume) 노드로 구성된 CC-NUMA 시스템 상에 상용 운영체제인 SCO UnixWare7을 기반으로 한 진단 소프트웨어를 개발함으로써 향후 개발될 다양한 중형 서버에 쉽게 이식할 수 있는 장점을 가지고 있다.

## 3. CC-NUMA 시스템을 위한 진단 메커니즘 설계

이 장에서는 현재 개발 중인 CC-NUMA 시스템을 위한 진단 메커니즘에 대한 설계 내용을 기술한다. 본 논문에서는 다중 노드로 구성된 CC-NUMA 구조를 위하여 소프트웨어 모듈들을 계층적으로 구성하였으며, 원격 진단 및 관리를 위한 클라이언트-서버 구조로 설계하였다. 이러한 설계 구조는 CC-NUMA 시스템에서 노드의 추가/삭제 등을 통한 확장성(scalability)을 보장할 뿐만 아니라, 인터넷을 통해 원격 관리가 가능하도록 지원한다.

### 3.1 진단 메커니즘 구조

설계된 진단 메커니즘은 크게 하드웨어 레벨(hardware level), 소프트웨어 레벨(software level), 및 사용자 인터페이스 레벨(user interface level)로 구분된다. 그림 2는 진단 메커니즘의 전체적인 구조를 나타낸다.

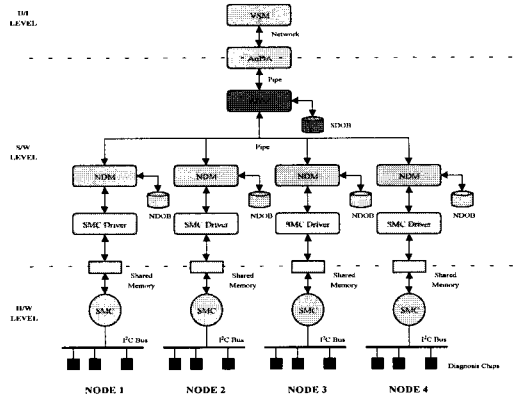



그림 2 진단 메커니즘의 계층적 구조

- 하드웨어 레벨(H/W level) : 진단 센서(diagnostic sensor), 진단 버스(diagnostic bus), 진단 칩셋(diagnostic chipset) 및 진단 정보 저장 메모리가 포함되며, 이들 진단 부품들은 각 노드마다 동일하게 장착되어 있다. 진단 센서로는 PCF8574, LM75, LM78 등의 디바이스가 사용되며, 주로 전압, 온도 및 팬 속도 등 부품의 상태를 읽어낸다. 진단 버스로는 $I^2C$ 버스[5]가 사용된다. 진단 칩셋은 진단 버스를 통해 진단 센서의 값을 읽고 이들의 정보를 관리한다. 또한, 다른 마이크로 컨트롤러와 진단 정보를 주고 받을 수 있는 부품이다. 이러한 부품으로는 80C652[6] 등의 마이크로 컨트롤러가 사용된다. 진단 정보를 저장하고, 호스트(host)와는 공유 메모리(shared memory) 인터페이스를 통해 정보를 교환한다.
- 소프트웨어 레벨(S/W level) : CC-NUMA 구조에 따라 각 노드로부터 수집된 진단 정보를 하나로 취합하여, 이를 사용자 인터페이스로 전달하는 역할을 담당한다. 이 레벨에는 진단 칩셋을 제어하기 위한 디바이스 드라이버(device driver), 각 노드의 진단 정보를 수집하기 위한 NDM(Node Diagnosis Manager), 시스템 전체적인 진단 정보를 관리하기 위한 SDM(System Diagnosis Manager) 및 사용자 인터페이스와의 통신 역할을 담당하는 AnDA(Administration and Diagnosis Arbitrator) 등의 소프트웨어 모듈이 있다. 또한, 진단 정보를 저장, 관리하기 위하여 SDM과 NDM은 각각 SDOB(System Diagnosed Objects Base)와 NDOB(Node Diagnosed Objects Base)를 데이타베이스 형태로 운용한다.
- 사용자 인터페이스 레벨(U/I level) : 시스템 관리자가 시스템을 진단하기 위하여 원격 진단 및 자바(Java) 기반 그래픽 사용자 인터페이스를 제공한다. 이 레벨에는 TCP/IP 프로토콜을 지원하기 위한 AnDA와 사용자 인터페이스 모듈인 VSM(Virtual System Manager)이 있다.

### 3.2 진단 소프트웨어 모듈

진단 메커니즘에서 소프트웨어 레벨을 담당하는 소프트웨어 모듈은 크게 SMC(System Management Controller), NDM(Node Diagnosis Manager), SDM(System Diagnosis Manager), AnDA(Adimistration & Diagnosis Arbitrator) 및 VSM(Virtual System Manager)으로 구성된다. 각 소프트웨어 모듈들의 기능을 요약하면 다음과 같다.

- SMC : 각 노드의 SHV(Standard High Volume)에 장착되는 마이크로 컨트롤러로서 노드 내의 진단 대상들에 대한 진단 정보를 수집하여, 이를 호스트에 전달하는 역할을 담당한다. 주로 진단 센서로 이루어진 진단 대상들과는 $I^2C$ 버스로 연결되어 있으며, 호스트와는 공유 메모리 인터페이스를 통해 데이타를 전달한다.
- NDM : 각 노드의 SMC를 주기적으로 폴링(polling)함으로써 진단 정보를 수집하고, 이를 NDOB에 저장한다. 또한, 이에 대한 변경이 발생하였을 때, 이를 SDM에 보고하는 역할을 수행한다. 또한, SDM으로터의 진단 정보 갱신이 발생하였을 때, 이를 SMC에 전달하는 역할을 수행한다.
- SDM : 여러 개의 노드로 구성된 시스템의 전체적인 진단 정보를 저장하고, 이를 관리한다. NDM으로부터의 정보 변경 내용이 보고되면, 이를 SDOB에 반영하며, 관리자로부터의 진단 정보 요청, 진단 정보 갱신 등의 기능을 수행한다.
- AnDA : 이것은 진단 모듈 및 관리 모듈과 사용자 인터페이스 모듈을 중개해주는 역할을 수행한다. 또한, 클라이언트-서버 환경에서 서버측의 통신 역

할을 전적으로 수행한다.

- VSM : 자바 기반 사용자 인터페이스를 제공하는 모듈로서, 서버와의 통신 및 편리한 사용을 위해 그래픽 및 트리 뷰(tree view)의 인터페이스를 제공해준다.

### 3.3 SDM 모듈 및 인터페이스

SDM의 목적은 기본적으로 여러 개의 노드로 구성된 시스템을 관리자에게 하나의 시스템으로 보일 수 있도록 단일 시스템 이미지(single system image)를 제공하는 것으로서 여러 진단 모듈 가운데 가장 핵심적인 모듈이다. SDM의 주요 역할은 다음과 같다.

- SDOB 초기화
- 각 NDM으로부터의 진단 정보 수집 및 SDOB 갱신
- AnDA로부터의 진단 정보 요청에 따른 응답
- AnDA로부터의 진단 정보 갱신 요청에 따라 해당 NDM으로의 갱신 요청

이를 위하여, SDM은 AnDA와 두 개의 메시지 채널, 각 NDM과 하나의 제어 채널, 그리고 모든 NDM이 공유하는 두 개의 채널을 가진다. 그림 3은 SDM 및 주변 모듈과의 인터페이스 구조를 나타내며, 네 개의 노드가 하나의 시스템을 구성하는 것을 가정하였다.

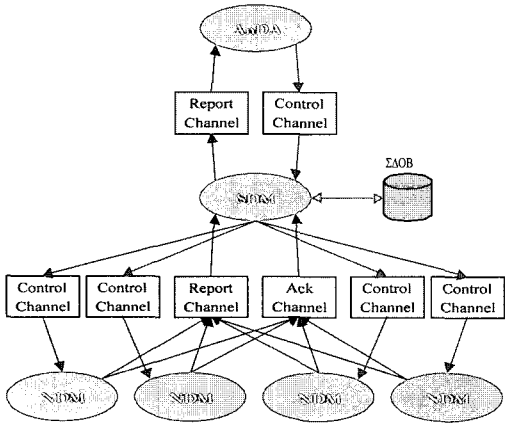



그림 3 SDM 모듈 인터페이스 구조

그림 3에서 보이는 바와 같이 SDM은 각 NDM에 대해 별도의 제어 채널(control channel)을 통해 제어 명령을 전달하며, 모든 NDM으로부터의 응답은 공유 채널인 보고 채널(report channel) 및 확인 채널(ack channel)을 통해 전달받는다. 반면, AnDA로부터의 요청은 제어 채널(control channel)을 통해 전달받으며, 이에 대한 응답은 보고 채널(report channel)을 통해 전달한다. 여기서 각 채널은 파이프(pipe)를 사용함으로써 FIFO(First-In First-Out) 기능을 제공한다. 또한, 시스템 전체적인 진단 정보는 시스템 진단 정보를 담고 있는 SDOB에 저장된다. SDOB에 대해서는 다음 절에서 자세히 설명하였다.

### 3.4 SDOB

SDOB는 시스템 전체적인 진단 정보를 저장하기 위한 저장소로서 SDM에 의해 관리된다. SDOB를 초기화하기 위해서는 DMI 2.0 사양의 MIF(Management Information Format) 형식의 텍스트 파일을 읽어들인다. 이 파일은 시스템을 구성하는 진단 대상들에 대한 정보를 그림 4와 같은 형식에 따라 표현한 것이다.

```
start group
                name = "group name"
//              class = "class string"
                [id = nnn]
//              [description = "description string"]
//              [key = nnn[,mm]…]
//              [pragma = "pragma string"]
                (여기에 속성 정의가 포함됨)
end group

start attribute
                name = "attribute name"
                id = nnn
//              [description = "description string"]
                type = datatype
                [access = method]
//              [pragma = "pragma string"]
//              [storage = storagetype]
                [value = {v|* "name"|"enum string"|unsupported|unknown}]
end attribute
```

그림 4 MIF 그룹 및 속성 정의

본 논문에서는 DMI 2.0 사양에서 제시된 MIF 형식 가운데 진단 정보를 표현하는데 필수적인 필드만을 기술하였다. 위에서 열거된 형식에서 // 로 시작되는 필드는 본 구현에서는 지원하지 않는 필드를 나타낸다.

MIF 파일을 기반으로 하여 SDM은 SDOB를 초기화하며, SDOB의 내부 자료구조도 마찬가지로 그룹 정보 및 속성 정보로 나누어지며, 각 자료구조의 필드는 그림 5와 같다.

```
typedef struct {
        char      name[MAX_ATTRNAME];
                                  /* attribute name */
        unsigned char     id;     /* attribute */
        accesstype_t  accesstype; /* access type */
        datatype_t    type;       /* attribute type */
        char      enumname[MAX_ENUMNAME];
                                  /* if type is enum */
        datavalue_t   value;      /* attribute value */
} attribute_t;

typedef struct {
        char      name[MAX_GROUPNAME];
                                  /* group name */
        unsigned char     id;     /* group id */
        ptr_grouplist             *next;
                              /* ptr to the next group */
        unsigned char             attrnum;
                          /* the number of attributes */
        attribute_t               *attr;
                           /* ptr to the attr array */
} group_t;
```

그림 5 SDOB 내부 자료의 구조체

그림 6은 그림 5의 자료구조를 그림으로 나타낸 것이다. 그림 6에서 보는 바와 같이, 각 그룹은 연결 리스트로 구성되어 있다. 또한, 하나의 그룹은 하나 이상의 속성 정보를 포함할 수 있는데, 이 때 각각의 속성들은 배열로 구성된다.

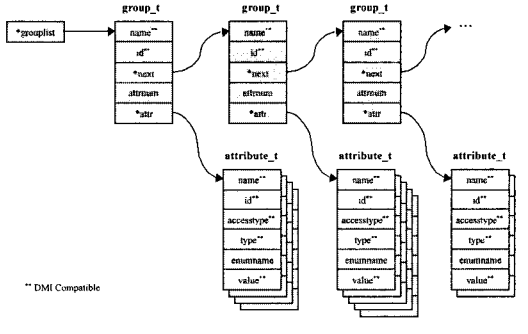



그림 6 SDOB 내부 자료 구조

그림 6에서 보는 바와 같이, 모든 그룹들은 grouplist라는 포인터를 시작으로 연결 리스트로 구성되어 있으며, 각 그룹은 attr 포인터를 통해 해당 그룹이 포함하고 있는 속성을 가리키고 있다. 또한, attrnum 필드는 속성의 개수를 나타내며, 이에 따라 속성 정보의 끝을 알 수 있다. 이러한 자료구조는 시스템 구성요소가 변경될 경우에도 동적으로 정보를 변경할 수 있도록 지원한다.

### 3.5 NDM-SDM 인터페이스

앞 절에서 언급된 대로, NDM과 SDM은 두 개의 공유 채널과 NDM 당 하나의 별도 채널을 통해 통신한다. 이들 채널에서는 모두 공통의 메시지 구조를 이용하는데 그 구조는 그림 7과 같다. 여기서 MAX_NDMMSGSIZE(=256)는 메시지의 최대 크기를 나타낸다.

```
typedef struct {
        unsigned char nid;        /* node id */
        unsigned char opcode;     /* op code */
        unsigned char bdysize;    /* body size */
        char          body[MAX_NDMMSGSIZE-3];
                                  /* message body */
} ndmmsg_t;
```

그림 7 NDM-SDM간 메시지 구조

또한, NDM과 SDM간의 프로토콜에서 각각의 채널 및 용도는 다음과 같다.

- 보고 채널(report channel) [모든 NDM → SDM] : 이 채널은 각 NDM으로부터 진단 정보를 수집할 때 사용된다. 각 NDM은 자체적으로 관리하고 있는 NDOB(Node Diagnosed Object Base)를 이용하여 SDOB의 갱신이 필요한 경우, 이 채널을 통해 ndmmsg(opcode = SYNC_SDOB)를 전송한다.
- 제어 채널(control channel) [SDM → 각 NDM]: 이 채널은 AnDA로부터 특정 노드의 진단 대상에 대한 갱신 요청이 SDM에 의해 수신된 경우, 이에 해당되는 노드에 갱신 요청이 포함된 ndmmsg(opcode = SYNC_NDOB)를 전송하기 위해 사용된다. 이 채널은 각 NDM에 대해 독립적으로 생성되어 있으므로 SDM은 노드 id에 따라 채널을 선택해야 한다.
- 확인 채널(ack channel) [모든 NDM → SDM] : 이 채널은 제어 채널을 통해 전달된 갱신 요청을 각 NDM이 수행한 이후, 이에 대한 결과가 포함된 ndmmsg(opcode = ACK_UPDATE)를 전송하기 위한 채널로서 모든 NDM이 공유하는 채널이다.

SDM이 제어 채널을 통해 NDM에 갱신 요청을 전송하면, 이에 대한 결과가 확인 채널을 통해 돌아올 때까지 블록(block)된다. 이러한 경우에도 NDM으로부터

의 SDOB 갱신 요청을 수신하기 위해서는 두 개의 공유 채널이 필요하다. 공유 채널인 경우, SDM은 ndmmsg의 nid 필드를 이용하여 메시지 송신자를 알아낼 수 있다

| Opcode | 이름 | 설명 |
|---|---|---|
| 0x00 | SYNC_NDOB | NDOB에 대한 갱신을 요청함 |
| 0x01 | SYNC_SDOB | SDOB에 대한 갱신을 요청함 |
| 0x11 | ACK_UPDATE | NDOB 갱신에 대한 확인 |

그림 8 NDM-SDM 간 프로토콜

### 3.6 AnDA-SDM 인터페이스

AnDA와 SDM은 두 개의 채널을 사용한다. 이들 채널에서 사용되는 메시지 구조는 그림 9와 같다. 이 구조체에서 identifier 필드는 AnDA에서 진단 모듈 및 관리 모듈을 구분하기 위한 것으로서 진단 모듈은 특정 값으로 고정되어 있다.

```
typedef struct {
        unsigned char identifier;  /* identifier */
        unsigned char opcode;      /* op code */
        unsigned int  bdysize;     /* body size */
        char          body[MAX_SDMMSGSIZE-3];
                                   /* message body */
} sdmmsg_t;
```

그림 9 AnDA-SDM 간 메시지 구조

AnDA와 SDM 간 인터페이스에서 사용되는 채널은 다음과 같다.

- 제어 채널(control channel) [AnDA → SDM]: 이 채널은 AnDA로부터의 진단 정보 요청 및 갱신 메시지를 수신하는데 사용된다. 이 경우, sdmmsg의 opcode는 그림 10과 같다.

| Opcode | 이름 | 설명 |
|---|---|---|
| 0x00 | REQ_CONNECT | 처음 연결을 설정함 |
| 0x01 | REQ_ATTRIBUTES | 특정 그룹의 속성을 요청함 |
| 0x02 | REQ_UPDATEATTRIBUTES | 특정 그룹의 속성을 갱신함 |
| 0x03 | REQ_GRAPH | 그래프 정보를 요청함 |
| 0xFF | REQ_DISCONNECT | 연결을 해제함 |

그림 10 제어 채널에서의 op code

- 보고 채널(report channel) [SDM → AnDA]: 이 채널은 AnDA로부터의 요청에 대한 응답 및 경보 메시지를 전송하기 위해 사용된다. 이 경우, sdmmsg의 opcode는 그림 11과 같다.

| Opcode | 이름 | 설명 |
|---|---|---|
| 0x10 | REP_ALLGROUPS | REQ_CONNECT에 대한 응답으로서 모든 그룹에 대한 정보를 보냄 |
| 0x11 | REP_ATTRIBUTES | REQ_ATTRIBUTES에 대한 응답으로서 해당 그룹의 속성 정보를 보냄 |
| 0x12 | REP_UPDATERESULT | REQ_UPDATEATTRIBUTES에 대한 응답으로서 갱신 결과를 보냄 |
| 0x13 | REP_GRAPH | REQ_GRAPH에 대한 응답으로서 그래프 정보를 보냄 |
| 0x20 | REP_ALERT | SDM에 의해 경보 메시지를 보냄 |

그림 11 보고 채널에서의 op code

### 3.7 AnDA-SDM 프로토콜

AnDA는 실제적으로 클라이언트와 접속하여 VSM과 직접 연결되기 때문에, AnDA와 SDM 간의 프로토콜은 클라이언트와 서버 간의 프로토콜이라 할 수 있다. 이러한 프로토콜은 크게 연결 설정, 연결 해제, 속성 읽기, 속성 갱신, 그래프 읽기 및 경보로 구성된다.

- 연결 설정(connection establishment) : 클라이언트 측에서 서버의 진단을 위해 처음 접속할 때 발생된다. AnDA는 가장 먼저 기존의 연결이 존재하고 있을 경우, 이를 해제하기 위하여 REQ_DISCONNECT메시지를 전송한다. 이후, 연결 설정을 위해 REQ_CONNECT를 전송한다. 이를 수신한 SDM은 SDOB(System Diagnosed Object Base)에 의해 관리되고 있는 모든 진단 대상-각 진단 대상은 SDOB 내에 DMI 형식에 따라 그룹(group)과 속성(attribute)들로 구성되어 있다-에 대한 명세(REP_ALLGROUPS)를 AnDA에게 전송해준다. AnDA에 연결된 VSM(Virtual System Manager)에 의해 모든 그룹에 대한 확인 작업이 종료되면, 계층적으로 구성된 트리 형식의 진단 대상들 중 루트(root)에 해당되는 그룹에 대

한 정보를 요청하기 위해 REQ_ATTRIBUTES 메시지를 전송한다. 이에 대한 응답으로 SDM은 루트 그룹을 SDOB에서 탐색하여 이에 대한 속성 정보가 담긴 REP_ATTRIBUTE 메시지를 AnDA에 보내는 것으로서 연결 설정 과정이 모두 종료된다.

- 연결 해제(connection tear-down) : 사용자 또는 다른 요인에 의해서 서버에 대한 진단 수행을 중지할 때 발생된다. 이 경우, AnDA는 연결 해제 메시지인 REQ_DISCONNECT 메시지를 SDM에 전송한다. SDM은 이에 대해 응답하지 않는다.
- 속성 읽기(get attribute) : 클라이언트 측에서 요구된 특정 그룹에 대한 정보를 요청할 때 발생된다. 이 경우, AnDA는 그룹 id가 명시된 속성 요청 메시지인 REQ_ATTRIBUTES를 SDM에 전송하며, SDM은 해당 그룹에 대한 속성 정보를 SDOB에서 검색하여 그 결과로서 REP_ATTRIBUTES를 AnDA에 전송해준다.
- 속성 갱신(update attribute) : 사용자가 사용자 인터페이스를 통해 특정 그룹에 대한 속성 정보를 갱신하고자 할 때 발생된다. AnDA는 그룹 id와 갱신된 속성 정보가 포함된 REQ_UPDATEATTRIBUTES 메시지를 SDM에 전송한다. SDM은 이 메시지에 따라 SDOB의 해당 속성을 갱신하고, 그 결과로서 REP_ UPDATERESULT 메시지를 AnDA에 전달한다.
- 그래프 읽기(get graph) : 팬 속도, 전압 및 온도 등에 대한 정보는 SDM에 의해 로깅된다. 이 프로토콜은 AnDA가 이에 대한 정보를 이용하여 그래프로 표현하고자 할 때 필요하다. AnDA는 먼저 해당 그룹의 REQ_GRAPH 메시지를 전송하고, SDM은 해당 그룹에 대한 로깅 DB를 검색하여 로깅 정보가 포함된 REP_GRAPH 메시지를 AnDA에 전송해준다.
- 경보(alert) : 이 프로토콜은 SDM이 능동적으로 AnDA에게 메시지를 보내기 위한 것이다. 이 프로토콜을 위하여 AnDA(또는 VSM)에서는 SDM으로의 핸드쉐이킹(handshaking) 없이도 SDM으로부터의 메시지를 수신할 준비가 항상 되어 있어야 한다. 즉, VSM은 SDM으로부터 경보 메시지를 수신한 다음, 이에 대한 어떠한 응답 메시지도 SDM에 보낼 필요가 없다는 의미이다. 따라서, VSM은 SDM으로부터 전달된 경보 메시지에 대하여 사용자 인터페이스를 통해 다양한 형태로 출력할 의무만을 가진다.

## 4. 구현 결과

### 4.1 구현 방법 및 실행 환경

CC-NUMA 시스템을 위한 진단 소프트웨어는 여러 개의 모듈로 구성되어 있으며, 이들 각각의 모듈들은 서로 다른 개발 환경에서 구현되었다. 먼저, 각 노드에서 동작하는 SMC 진단 컨트롤러는 현재 시스템이 개발되기 전이기 때문에 SMC 에뮬레이터를 작성하였다. 이 에뮬레이터는 UnixWare7 상에서 C로 개발하였다. 또한, NDM, SDM, AnDA 등의 모듈도 마찬가지로 UnixWare7 상에서 C로 개발하였다. 반면, 자바 애플리케이션(Java application)인 VSM은 Windows 98 상에서 JDK 1.1.6을 지원하는 J-Builder 2.0을 사용하여 개발하였다.

구현된 진단 소프트웨어의 실행 환경은 다음과 같다. 먼저, CC-NUMA 시스템이 현재 개발 중에 있으므로, 단일 SMP 시스템 내에서 각 노드를 가상으로 설정하였다. 즉, 두 개의 노드를 가정하였을 경우, 두 개의 SMC 에뮬레이터 및 두 개의 NDM이 동작한다. 반면, SDM 및 AnDA는 각각 하나의 프로세스로서 동작한다. 실제로, 구현된 진단 소프트웨어는 UnixWare 7이 탑재된 서버에 SMC 에뮬레이터, NDM, SDM 및 AnDA가 데몬(daemon)으로 수행되며, 이와 네트워크로 연결된 Windows 98 클라이언트 PC에 VSM 자바 애플리케이션을 통해 시스템을 진단한다. 또한, 관리자의 서버 접속시 보안을 위하여 VSM에는 암호의 복호 기능이 구현되어 있다.

### 4.2 구현 결과

이 절에서는 본 논문에서 구현된 진단 소프트웨어의 실행 화면을 소개한다.

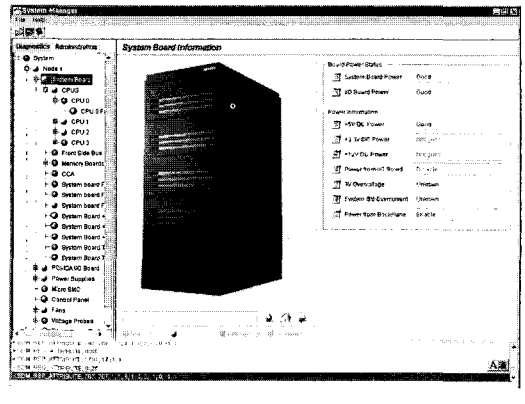



그림 12 진단 소프트웨어의 실행 화면

그림 12는 구현된 진단 소프트웨어의 실행 화면을 나타내며, 크게 다음과 같은 윈도우(window)로 구성되어 있다.

- 트리 윈도우(tree window) : 시스템을 구성하는 디바이스들을 물리적 및 논리적 위치에 따라 트리 형태로 구성한 윈도우이다. 관리자는 트리를 브라우징(browsing)함으로써 해당 디바이스에 쉽게 접근할 수 있다.
- 이미지 윈도우(image window) : 시스템의 이미지가 출력되는 윈도우로서, 관리자는 시스템 이미지를 이용하여 해당 디바이스에 접근할 수 있다.
- 정보 윈도우(information window) : 관리자에 의해 선택된 디바이스의 진단 정보를 나타내는 윈도우이다.
- 상태 윈도우(status window) : 이미지 윈도우 하단에 선택된 디바이스의 상태를 나타내는 윈도우이다.
- 로그 윈도우(log window) : VSM과 AnDA 간 송수신되는 내용을 포함한 로그 내용을 출력하는 윈도우이다.

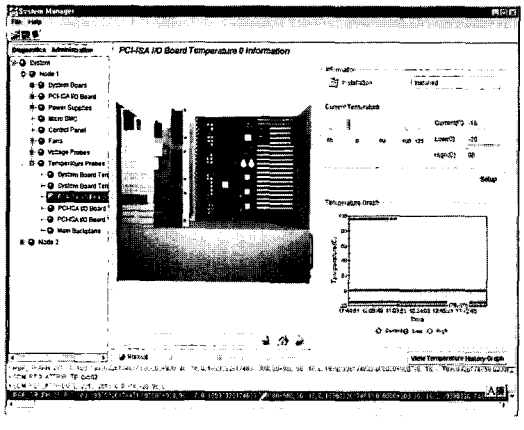


그림 13 디바이스 정보 및 그래프 출력 화면

그림 13은 실제 디바이스에 대한 진단 내용을 출력한 화면을 보여준다. 이 그림에서 디바이스의 진단 속성에 대한 정보와 함께 정보 히스토리(history) 등을 그래프로 출력한다. 그림 13은 시스템 디바이스 중 PCI-ISA I/O 보드의 첫번째 온도 센서로부터 얻어진 정보, 즉 현재 온도, 최저 임계치, 최고 임계치 등의 온도 정보와 그래프를 나나탠다.

### 4.3 주요 기능

이 절에서는 본 논문에서 구현된 진단 소프트웨어의 주요 기능을 요약하였다. 이러한 모든 기능들은 원격에 위치한 관리자가 인터넷을 통해 원격 제어가 가능하다. 인터넷을 통해 서버에 접속하는 관리자는 서버의 루트(root) 암호를 통해 인증되며, 한 순간 한 명의 관리자만이 접속할 수 있다. 또한, 관리자에 의해 사전에 설정된 주요 진단 파라미터들의 상한 및 하한 임계치에 대해 진단 소프트웨어는 모니터링된 값이 그 임계치를 벗어날 경우, 시스템 경보 기능을 작동시킬 수 있다.

다음은 진단 소프트웨어의 주요 기능을 나타낸다.

- 시스템 상태 모니터링 기능 : 시스템의 상태를 컬러 아이콘으로 표시함으로써 관리자가 현재 시스템의 상태를 즉시 파악할 수 있도록 제공한다. 또한, 이를 이용하여 문제가 발생한 부분으로 바로 이동할 수 있는 기능을 제공한다. 각 부품의 상태는 정보의 속성에 따라 텍스트, 이미지 또는 그래프로서 표시된다.
- 임계치 등 시스템 상수 세팅 기능 : 온도, 전압 및 팬의 속도 등 동작 범위가 주어져 있는 항목에 대해서는 그에 대한 임계치 및 시스템 상수를 지정할 수 있는 기능을 제공한다.
- 히스토리(history) 및 메시지 로깅(logging) 기능 : 관리자가 접속하지 않은 상태일지라도 진단 소프트웨어에 의해 검출되는 진단 정보 및 오류 메시지 등은 내부 로그 파일(log file)에 로깅된다. 또한, 온도, 전압 및 팬 속도 등 가변적인 값을 가지는 항목들은 값이 변경될 때마다 히스토리 파일(history file)에 기록된다. 이러한 정보는 추후 관리자의 시스템의 상태 파악에 도움을 주게 된다.
- 시스템 제어 기능 : 시스템의 상태에 따라 시스템 다운(system down), 웜 리셋(warm reset) 및 콜드 리셋(cold reset) 등의 기능을 제공한다. 이러한 기능은 관리자에 의해 수동적으로 행해질 수도 있지만, 진단 소프트웨어의 자체 진단 결과에 의해 시스템이 불안정한 상태가 지속될 경우 자동적으로 수행될 수도 있다. 이러한, 진단 파라미터의 값들은 사전에 관리자에 의해 세팅되어야 한다.

## 5. 결 론

중형 서버 시장에서 서버의 가용성 및 확장성은 매우 중요한 요소라 할 수 있다. 이러한 관점으로 볼 때, CC-NUMA 기술을 통해 기존 SMP 서버의 확장성을

향상시킴으로써 중형 서버 시장을 보다 확대할 수 있는 계기가 되고 있다. 그러나, 이러한 CC-NUMA 기술은 하드웨어적으로는 여러 개의 노드가 묶여 있지만, 소프트웨어적으로는 하나의 시스템으로 인식되어야 하기 때문에 기존의 시스템 관리 및 진단 소프트웨어를 적용하기 매우 어렵다. 즉, 각 노드를 별개의 시스템으로 인식하거나, 하나의 노드만을 인식하는 방법만이 가능하다.

이러한 CC-NUMA 구조의 특성을 지원하기 위해 본 논문에서 제시한 설계 목표 및 해결 방안은 다음과 같다.

첫째, CC-NUMA 구조의 특성, 즉 여러 개의 노드들이 하나의 시스템으로서 동작하는 구조에 대해 하드웨어적으로나 소프트웨어적으로 적합한 메커니즘을 제공한다. 이를 위하여, 본 논문에서는 각 노드마다 NDM이라는 노드 진단 매니저를 두고, 이를 통합하기 위하여 SDM이라는 시스템 진단 매니저를 두는 계층적 구조로 설계하였으며, 이들이 각각의 데이타베이스를 운용함으로써 모듈간 인터페이스를 최소화하였다. 이러한 모듈 구조는 노드의 개수가 증가 또는 감소하더라도 소프트웨어 모듈 구조를 변경할 필요가 없으므로 높은 확장성을 보장한다.

둘째, TCO(Total Cost of Ownership)를 절감하기 위한 방안으로서 원격 진단 및 관리 기능을 제공한다. 이를 위해 관리자가 원격 클라이언트인 PC 상에서 서버에 접속할 수 있도록 네크워크를 통한 인터페이스를 제공한다. 또한, 클라이언트 시스템에 비의존적인 자바 애플리케이션을 이용한 사용자 인터페이스를 통해, 관리자가 어느 시스템에서도 서버에 접근할 수 있도록 접근성을 향상시켰다.

셋째, 상용 운영체제인 SCO UnixWare7을 기반으로 한 소프트웨어 모듈을 구현함으로써 특정 시스템에 국한되지 않고 범용적인 시스템에 이식할 수 있도록 하였다. 즉, 표준화된 SMP 노드인 SHV(Standard High Volume)를 이용하여 CC-NUMA 시스템을 구성하였을 경우, 이러한 시스템에 쉽게 이식할 수 있는 장점을 가진다.

결과적으로, 본 논문에서 개발된 CC-NUMA 시스템을 위한 진단 시스템은 기존의 일반적인 진단 및 관리 소프트웨어의 범위에서 벗어난 시스템 구조를 지원할 뿐만 아니라, 원격 진단 및 관리 기능을 보강함으로써 실제적으로 서버를 운용하는데 있어 많은 도움을 줄 것으로 예상된다.

본 연구에서는 현재 IPMI 인터페이스 지원, 오프-라인(off-line) 진단 기능을 수행하기 위한 진단 보드(diagnostic board) 개발을 수행하고 있다. 진단 보드가 각 노드에 장착될 경우, 관리자는 시스템이 다운(down)되어 있는 경우에도 서버에 접근 시스템의 상태를 검사할 수 있으며, 이를 이용하여 리셋(reset), 리부팅(rebooting) 등의 조치를 취할 수 있게 될 것이다.

## 참 고 문 헌


[1] Intel, Wired for Management Baseline Version 2.0, December 1998.

[2] DMTF, Desktop Management Interface Specification Version 2.00, March 1996.

[3] Intel, HP, NEC, and Dell, Intelligent Platform Management Interface Specification v1.0, September 1998.

[4] DMTF, Web-Based Enterprise Management, from http://www.dmtf.org/wbem/

[5] Philips, The I2C-Bus Specification Version 2.0, December 1998.

[6] Philips, 80C652/83C652 CMOS single-chip 8-bit microcontrollers Product Specification, August 1996.

[7] D. Culler, J. P. Singh and A. Gupta, Parallel Computer Architecture; A Hardware/Software Approach, Morgan Kaufmann Publishers, August 1997.

[8] Intel, Intel Server Control Installation and User's Guide, May 1998.

[9] SCO, SCO UnixWare System Administration, Feb 1996.

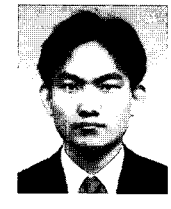
[10] Computer Associates, Unicenter TNG, from http://www.cai.com/

[11] Hewlett Packard, OpenView NNM/NT, from http://www.hp.com/

[12] Intel, LANDesk Server Manager Pro v6.0, from http://www.intel.com/

[13] Data General, M3D (Monitor, Manage, and Maintain) software, from http://www.dg.com/

[14] Sequent Computer Systems, VCS (Virtual Console Software), from http://www.sequent.com/


정 태 일

1992년 중앙대학교 전자계산학과 공학사. 1994년 중앙대학교 전자계산학과 공학석사. 1994년 ~ 1998년 중앙대학교 컴퓨터공학과 공학박사. 1998년 ~ 현재 한국전자통신연구원 근무. 관심분야는 멀티미디어 운영체제, 멀티미디어 통신임.

정 낙 주

1992년 충남대학교 컴퓨터공학과(공학사). 1995년 충남대학교 컴퓨터공학과(공학석사). 1995년 ~ 현재 한국전자통신연구원 리눅스연구팀. 관심분야는 운영체제, 시스템 관리

김 주 만

1984년 숭실대학교 전자계산학과 (공학사). 1998년 충남대학교 대학원 컴퓨터공학과(공학석사). 1995 ~ 1996년 미국 노벨 및 SCO사 국제공동연구 파견근무. 1985년 ~ 현재 한국전자통신연구원 운영체제연구팀 팀장(책임연구원). 1994년 정보처리 기술사(전자 계산기 조직 응용). 관심분야는 운영체제, 병렬 처리, 결함 허용 시스템

김 해 진

1983년 경북대학교 컴퓨터공학과(학사). 1995년 충남대학교 전산학과(석사). 1998년 한국정보통신대학원대학교 박사 과정. 1992년 정보처리기술사(전자계산조직응용). 1983년 3월 ~ 현재 한국전자통신연구원 운영체제연구팀(책임연구원). 관심분야는 운영체제, 병렬처리, 실시간 시스템, 고장감내