# 클러스터 기반 DBMS를 위한 고가용성 클러스터 관리기의 설계 및 구현

## (Design and Implementation of High-Availability Cluster Manager for Cluster-based DBMS)

김 영 창 † 장 재 우 †† 김 홍 연 †††
(Young-Chang Kim) (Jae-Woo Chang) (Hong-Yeon Kim)

**요 약** 최근 컴퓨터에 의해 처리되어야 할 정보의 양이 급증하면서, 이의 처리를 위해 여러 개의 단일 서버를 고속의 네트워크(network)으로 연결한 클러스터 컴퓨팅 시스템 (*cluster computing system*)이 등장하게 되었다. 그 결과, 클러스터 기반 DBMS에 관한 연구가 국내외적으로 활발히 진행 중이며, 이에 따라 클러스터 기반 DBMS를 모니터링 및 관리하는 클러스터 관리기의 개발이 요구된다. 그러나, 클러스터 기반 DBMS를 효율적으로 관리할 수 있는 관리 도구에 대한 연구는 미흡한 실정이다. 따라서 본 논문에서는 클러스터 기반 DBMS를 위한 고가용성 클러스터 관리기를 설계 및 구현한다. 구현된 클러스터 관리기는 그래픽 사용자 인터페이스(GUI)를 통해, 클러스터 시스템 내의 노드의 상태 및 각 노드에서의 DBMS 상태를 모니터(monitor)한다. 아울러, 각 서버의 상태를 모니터한 정보를 바탕으로, 서버의 오류를 감지하고 복구함으로써 클러스터 기반 DBMS의 고가용성을 지원한다.

**키워드** : 클러스터 DBMS, 고가용성, 클러스터 관리기

***Abstract*** As the amount of information to be processed by computers has recently been increased, there has been cluster computing systems developed by connecting PCs and workstations using high-speed networks. As a result, the studies on a cluster-based DBMS have been done with a wide range and it is necessary to develop a *cluster manager for monitoring and managing cluster-based* DBMSs. But, a cluster manager for efficiently managing cluster-based DBMSs has been studied in a first step. In this paper, we design and implement a high-availability cluster manager for a cluster-based DBMS. It monitors the status of nodes in cluster systems as well as the status of DBMS instances in a node, by using a graphic user interface (GUI). Our cluster manager supports the high-availability for a cluster-based DBMS by perceiving errors and recovering them, based on the monitored information on the status of a server.

**Key words** : cluster DBMS, high-availability, cluster manager

## 1. 서 론

최근 인터넷 환경에서 급속히 증대되는 24시간 무정지 서비스 요구를 효과적으로 처리하기 위하여, 저비용으로 시스템 성능 및 시스템 확장을 용이하게 하는 클러스터 컴퓨팅 시스템(cluster computing system)이 필요하게 되었다[1,2]. 그 이유는 이러한 인터넷 환경에서 기존의 단일 대용량 데이타베이스 서버를 사용하여 고성능(*high performance*)과 고가용성(*high availability*)의 서비스를 제공하는 것이 한계에 도달하였기 때문이다. 이러한 한계를 극복하고 더욱 강력한 컴퓨팅 파워와 시스템의 안정적 서비스를 제공하기 위해, 클러스터 기반 DBMS에 대한 연구 및 개발이 활발히 이루어지고 있다. 실제로 Oracle 9i Real Application Server, Informix Extended Parallel Server, IBM DB2 Universal Database EEE 등은 이러한 클러스터 기반 DBMS 이다. 이러한 클러스터 기반 DBMS는 대규모 데이타를 여러 노드에 분산 저장하고 일관성 있게 접근할 수 있는 메커니즘을 제공하며, 또한 클러스터 시스템의 특징인 고성능, 고가용성, 고확장성(high scalability)

† 학생회원 : 전북대학교 컴퓨터공학과
yckim@dblab.chonbuk.ac.kr
†† 종신회원 : 전북대학교 컴퓨터공학과 교수
jwchang@chonbuk.ac.kr
††† 정 회 원 : 한국전자통신연구원 데이타베이스연구팀 연구원
hykim@etri.re.kr
논문접수 : 2005년 4월 7일
심사완료 : 2005년 10월 26일

을 지닌다[3-5].

이러한 클러스터 기반 DBMS를 효율적으로 관리하기 위해서는 다음과 같은 기능을 지니는 관리 도구가 필요하다. 첫째, 다수의 노드로 구성된 클러스터 시스템을 단일 시스템처럼 인식할 수 있는 환경이 제공되어야 한다. 즉, 시스템 내의 어떠한 노드에서도 클러스터 내의 모든 시스템 자원과 행위를 제공받을 수 있어야 한다. 둘째, 전체 시스템 구성과 각 노드의 부하 및 CPU, 메모리, 디스크 등 자원의 활용상태의 파악이 용이하여야 한다. 셋째, 모든 노드의 성능을 최대한 발휘하기 위해 사용자의 요구를 적절히 분산시키는 스케줄링 방법이 필요하다. 마지막으로, 고가용성을 위해 노드 장애가 발생한 상황에 대응할 수 있는 장애극복(fail-over)방법이 필요하다[6,7].

이를 위해 본 논문에서는 클러스터 기반 DBMS를 위한 고가용성 클러스터 관리기를 설계 및 구현한다. 이는 각 노드들에 대해서 IP(Internet Protocol) 레벨의 단일 시스템 이미지를 제공함으로써, 사용자의 트랜잭션이 어느 노드에서 수행되는 지에 구애를 받지 않는 투명성(transparency)를 제공한다. 즉, 클러스터를 구성하는 모든 서버는 하나의 공통된 IP 주소를 공유하기 때문에, 서비스를 요청하는 사용자 패킷은 서버와 관리노드의 역할을 동시에 수행하는 노드를 거쳐 적절한 서버노드에서 서비스를 받는다. 또한 각 노드에서 모니터(monitor) probe 가 시스템의 자원 상태와 데이타베이스 인스턴스(instance)를 모니터하고, 해당 서비스의 오류를 감지하여 이를 관리노드에 전달한다. 관리노드는 전체 시스템의 네트워크 오류를 감지하고, 고장이 발생하면 모니터 probe 로부터 받은 정보를 바탕으로 이에 따른 적절한 장애극복 절차를 수행함으로써, 전체 클러스터 기반 DBMS가 정상적인 서비스를 수행할 수 있도록 지원한다. 아울러 클러스터 관리기를 토대로 각 서버노드의 시스템 자원 활용 상태 및 DBMS 인스턴스 상태를 나타내고, 또한 사용자가 클러스터 기반 DBMS의 관리를 용이하도록 지원한다.

본 논문의 구성은 다음과 같다. 제2장에서는 관련연구로서 Oracle Parallel Server의 관리 도구로 사용되는 OCMS 및 리눅스 Beoulf 클러스터 시스템을 위한 관리 소프트웨어인 SCMS에 관해 소개한다. 제3장에서는 본 논문에서 설계한 클러스터 기반 DBMS 구조 및 클러스터 관리기에 대해 기술하고, 제4장에서는 설계된 클러스터 관리기 및 사용자 인터페이스를 iBASE/Cluster를 이용하여 구현 및 테스팅을 수행하고, 마지막으로 제5장에서는 결론 및 향후연구를 제시한다.

## 2. 관련 연구

본 절에서는 클러스터 기반 DBMS를 위한 고가용성 클러스터 관리기를 설계하기 위해 기존의 관리도구를 살펴본다. 이를 위해 DBMS를 위한 관리도구로 가장 널리 알려진 오라클 8i의 OCMS(Oracle Cluster Management Software) 및 리눅스 Beowulf 클러스터 시스템을 위한 관리 소프트웨어인 SCMS(SMILE Cluster Management System)를 살펴본다.

### 2.1 OCMS(Oracle Cluster Management Software)

대표적으로 알려진 클러스터 DBMS용 관리도구인 OCMS[8]는 오라클에서 리눅스용 오라클 8i Enterprise Edition에 번들로 제공하는 클러스터 DBMS용 관리도구로서 클러스터 멤버쉽(membership) 서비스, 클러스터 전체의 일관된 모습, 노드 모니터링, 클러스터 재설정의 서비스를 제공한다. OCMS는 리눅스용 오라클 8i Parallel Server를 선택했을 때 자동으로 설치되는 컴포넌트로써, watchdog 데몬(daemon), 노드 모니터(node monitor), 클러스터 관리자(cluster manager)의 3개 컴포넌트를 통해 오라클이 클러스터 시스템 내에서 동작할 수 있도록 한다. 그림 1은 OCMS의 시스템 구조를 나타낸다. 첫째, 오라클에서 개발한 watchdog 데몬은 데이타베이스의 오류를 방지하기 위해, 시스템 자원을 모니터하는 리눅스의 watchdog 타이머(timer)를 사용한다. watchdog 데몬은 노드 모니터와 클러스터 관리자를 모니터하며, 정해진 시간간격으로 watchdog 타이머에게 메시지를 전달한다. watchdog 타이머는 시스템이 다운되었을 때 데이타베이스의 오류를 방지하기 위해 서버를 초기화한다.

한편, 노드 모니터는 클러스터의 일관된 뷰(consistent view)를 관리하고 클러스터 관리자에게 각 노드의 상태 정보를 전달한다. 노드 모니터는 클러스터에 있는 모든 노드들의 상태정보를 데이타베이스에서 관리한다. 노드 모니터는 정해진 시간 간격 안에 ping 메시지에 대한 응답을 받지 못하면 해당노드를 비활성(inactive) 상태로 표시한다. 노드 모니터에 의해 탐지되는 노드의 오류는 원격 노드의 노드 모니터가 종료되었을 경우, 네트워크 오류가 일어났을 경우, 그리고 원격 노드가 과부하 상태일 경우로 간주될 수 있다. 노드 모니터는 특정 디스크 영역을 사용해서 노드와 네트워크의 오류를 구별한다. 만약 클러스터 시스템내의 노드가 공유디스크에 기록을 계속하면서도 ping 메시지에 응답하지 않으면, 네트워크 오류로 간주하고 해당노드를 종료시킨다. 노드가 과부하 상태일 경우에는 watchdog 데몬을 통해 해당노드로 더 이상 부하가 걸리지 않도록 한다. 마지막으로, 클러스터 관리자는 프로세스 레벨에서의 클러스터 시스템의 상태를 관리한다. 클러스터 관리자는 클러스터 시스템 내에 오라클 인스턴스의 등록을 받고, 인스턴스

에게 상태정보를 전달해 인스턴스 간의 통신을 수행한다. 만약 오라클 인스턴스가 디스크에 비정상적인 기록을 하고자 시도할 때, 클러스터 관리자는 데이타베이스 오류를 방지하기 위해 해당 인스턴스가 공유디스크에 기록하지 못하도록 watchdog 데몬에게 통지하고 이를 멈추게 한다. 그러나, OCMS는 Oracle이라는 특정 DBMS를 위한 관리도구로서 다른 DBMS를 위한 관리도구로 쓸 수 없다는 단점이 존재한다. 아울러, 클러스터 관리자가 각 노드마다 분산되어 서로 통신을 통해 시스템을 관리하기 때문에 네트워크 자원을 효율적으로 이용하지 못한다. 또한 공유 디스크에 quorum 파티션을 설정하여 사용하여 다른 서버 노드의 오류를 감지한다. 이러한 두 특징으로인해 노드의 수가 증가할수록 전체 시스템의 성능 저하되는 단점이 존재한다.

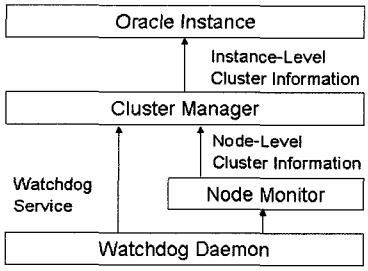



그림 1 OCMS의 시스템 구조

### 2.2 SCMS(SMILE Cluster Management System)

SCMS[9]는 리눅스 Beowulf 클러스터 시스템을 위한 관리 소프트웨어로 Tailand의 Kasetsart 대학에서 개발하였다. SCMS는 Beowulf 클러스터를 위한 시스템 관리도구를 제공하고, 이를 통해 노드의 종료 및 재시작, 노드로의 원격 접속과 노드에서의 명령어 실행, 클러스터 환경에서 동작하는 병렬처리 명령어 제공, 노드의 환경설정, 각 노드나 클러스터 전체의 CPU, 메모리, I/O, 네트워크 사용량을 모니터한다. SCMS의 시스템 구조는 그림 2에 나타나며, CMA(Control and Monitoring Agent), SMA(Systems Management Agent), RMI (Resource Management Interface)의 3개 컴포넌트로 이루어진다.

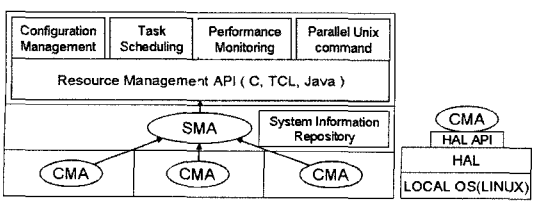



그림 2 SCMS의 시스템 구조

SCMS는 각 노드에서 시스템의 상태 정보를 모니터하는 컴포넌트로서, HAL(Hardware Abstraction Layer)이라 불리는 층을 통해 서로 다른 플랫폼에서도 동작할 수 있도록 한다. SMA는 CMA로부터 각 노드들의 상태정보를 모아 저장하고, 사용자의 요구를 CMA에게 전달하여 노드가 사용자의 명령을 처리할 수 있게 한다. 하나의 SMA가 수용할 수 있는 노드의 수에는 제한이 있으며, 노드의 수가 많아질 경우 SMA를 단계적으로 구성함으로써 이를 해결한다. RMI는 SMA가 가진 노드들의 정보를 바탕으로 동작하는 상위 애플리케이션을 위한 API(Application Program Interface)들의 집합이다. 이를 통해 각 노드 및 클러스터의 환경을 설정하는 프로그램, 각 노드의 상태를 확인할 수 있는 모니터링 소프트웨어 및 스케줄러, 클러스터 환경에서 하나의 노드에서처럼 동작할 수 있는 유닉스 명령어들을 제공한다. 그러나, SCMS는 클러스터 시스템을 위한 관리기로 클러스터 기반 DBMS의 오류 감지 및 회복 절차를 지원하지 못하는 단점이 있다.

## 3. 클러스터 기반 DBMS를 위한 고가용성 클러스터 관리기의 설계

클러스터 기반 DBMS를 위한 고가용성 클러스터 관리기는 각 서버노드의 시스템 자원과 데이타베이스를 모니터하여, 모니터된 정보를 관리노드에 전송함으로써 사용자가 현재 서버의 상태를 정확히 파악할 수 있게 하며, 또한 이를 바탕으로 각 서버의 시스템과 데이타베이스의 오류를 감지하고 복구함으로써, 전체 클러스터 기반 DBMS가 정상적인 서비스를 수행할 수 있도록 지원한다.

### 3.1 클러스터 기반 DBMS의 구조

클러스터 기반 DBMS는 여러 개의 서버노드가 디스크를 공유하는 환경이며, 서버노드는 고속의 기가비트 이더넷을 통해 내부 네트워크로 연결되어 있으며, 서버노드는 OS로 리눅스(RedHat 7.1)를 사용한다. 서버노드 중의 한 노드는 데이타베이스 서버의 역할과 전체 클러스터 시스템의 게이트웨이(gateway) 역할을 동시에 수행한다. 아울러 리눅스 가상 서버를 이용하여 사용자의 서비스 요청을 스케줄링 알고리즘에 따라 적절한 서버에 할당하여 처리하며, 또한 각 서버의 모니터된 정보를 통합 관리하는 관리노드의 역할을 수행한다. 클라이언트의 요구는 가상 IP로 관리노드에 전달되고, 관리노드는 요청받은 사용자의 서비스 요청을 리눅스 가상 서버의 스케줄링 알고리즘에 따라 선택된 적절한 서버노드로 서비스 요청을 재분배한다. 사용자의 서비스 요청을 관리노드로부터 전달받은 서버노드는 이를 처리하여 사용자에게 결과를 전송한다. 이때, 관리노드에 오류가 발생

하면 사용자의 서비스 요청이 서버노드로 전달될 수 없기 때문에, 전체 클러스터 기반 DBMS가 서비스를 수행할 수 없다. 이를 위해 본 논문에서는 관리노드 이외의 서버노드 가운데 한 노드를 데이타베이스 서버의 역할과 백업관리노드의 역할을 동시에 수행하도록 지정한다. 마찬가지로 백업관리노드에서 오류가 발생하면, 관리노드를 제외한 서비스 가능한 서버노드중의 하나가 백업관리노드의 역할을 수행한다. 따라서 전체 클러스터 기반 DBMS는 모든 서버노드가 데이타베이스 서버의 역할을 수행하며, 서버중의 한 노드는 관리노드의 역할을, 다른 하나의 노드는 백업관리노드의 역할을 동시에 수행한다. 본 논문에서 사용하는 클러스터 기반 DBMS의 구성도는 그림 3과 같다.

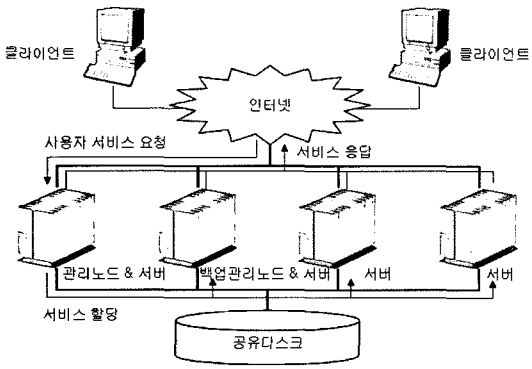



그림 3 클러스터 기반 DBMS의 구성도

### 3.2 클러스터 관리기

클러스터 기반 DBMS 환경에서 클러스터 관리기는 부하 및 각 자원의 상태를 정확히 모니터하여 각 서버 및 데이타베이스 오류를 감지하고, 사용자에게 신뢰성(reliability)있는 정보를 제공하며, 다른 애플리케이션이 이런 상태정보에 접근할 수 있는 API를 제공한다. 이러한 클러스터 관리기를 설계함에 있어 고려해야 할 사항은 다음과 같다. 첫째, 모니터 대상을 최소화하여 노드의 부하를 가중시키는 것을 방지한다. 둘째, 모니터 주기(frequency)를 동적으로 변화시켜, 노드의 상태정보를 관리노드로 전송할 때 발생하는 네트워크 전송(traffic)량을 적절히 조절한다[11].

클러스터 관리기는 CM(Cluster Manager), NM(Node Manager), probe, handler의 네 개의 컴포넌트로 이루어지며, 그림 4는 클러스터 관리기의 컴포넌트 구성도를 나타낸다. 여기서 probe와 handler는 각각 시스템, DB, LB(Load Banlancer) probe와 handler로 분류된다.

첫째, 시스템, DB, LB probe는 각 서버에서 CPU, 메모리, 네트워크의 시스템 상태, DBMS 인스턴스 상태,

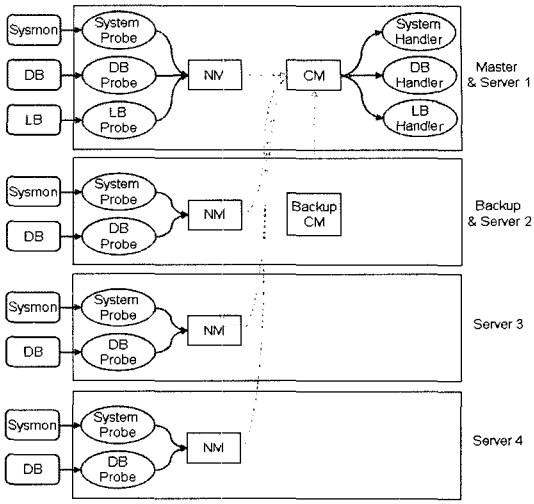



그림 4 클러스터 관리기의 컴포넌트 구성도

그리고 부하분산기를 각각 모니터한다. 부하분산기로는 리눅스 가상 서버를 사용한다[10]. 리눅스 가상 서버가 지원하는 시스템의 구조 중 direct routing 방식을 사용하며, 스케줄링(scheduling) 알고리즘으로는 round robin 방법을 사용한다. 각각의 probe는 모니터되는 대상 서비스의 상태에 따라 이벤트를 발생시킨다. 모니터를 시작 및 종료할 때, 각각 이벤트를 발생시키고 이를 서비스 상태 테이블에 등록한다. 정상동작중일 경우에는 모니터된 정보를 이벤트(ALIVE 이벤트)와 함께 각 handler에게 전달한다. 아울러 모니터 대상 서비스의 오류가 발생하였을 경우 시스템 handler는 probe를 재시작하고 네트워크에 오류가 발생하였을 경우 이를 서비스 상태 테이블에 등록한다. DB handler는 수행중인 트랜잭션의 복구를 위해 정해진 복구 절차를 수행하고, 해당 노드로 사용자의 서비스 요청이 전달되지 않도록 부하분산 대상에서 제거한다. LB handler는 해당 프로세스를 원격으로 재시작 시킨다.

둘째, NM은 관리노드에 존재하는 CM과 클러스터 시스템의 각 서버노드간의 네트워크 통신을 담당하는 컴포넌트이다. 표 1은 NM이 발생시키는 이벤트와 이에 따른 수행절차를 나타낸다.

NM은 시작 및 종료 시 이벤트를 발생하여 CM이 서비스 상태 테이블을 갱신하도록 한다. 정상동작 시는 각 서비스별 probe로부터 전달받은 정보와 이벤트를 CM에게 전달하고, CM으로부터의 응답을 통해 CM의 정상동작상태를 모니터한다. 만약 정해진 시간동안 응답이 없을 경우 CM의 오류로 간주하고 백업관리노드의 CM으로 네트워크 연결을 재설정한다.

마지막으로, CM은 관리노드에 위치하며 NM으로부터

표 1 NM이 발생시키는 이벤트와 이에 따른 수행절차

| Event Source | Status | Event | Procedure |
|---|---|---|---|
| NM | Start | NM 시작 | 서비스 상태 테이블에 등록 |
| NM | Run | NM 정상동작 | 노드 정보를 CM에 전달 |
| NM | Stop | NM 종료 | 서비스 상태 테이블에 등록 |
| NM | Error | CM 오류 | 백업관리노드의 CM으로 연결 재설정 |

표 2 CM이 발생시키는 이벤트와 이에 따른 수행절차

| Event Source | Status | Event | procedure |
|---|---|---|---|
| CM | Error | System Probe 오류 발생 | System handler에게 전달 |
| CM | Error | DB Probe 오류 발생 | DB handler에게 전달 |
| CM | Error | LB Probe 오류 발생 | LB handler에게 전달 |
| CM | Error | NM 오류 발생 | System handler에게 전달 |
| CM | Error | 서비스 네트워크 고장 | 각 서비스 handler에게 전달 |
| CM | Error | 클러스터 네트워크 고장 | 각 서비스 handler에게 전달 |
| CM | Error | 노드 완전 고장 | 각 서비스 handler에게 전달 |
| CM | Error | 백업 CM 오류 | 서비스 가능 노드에 백업 CM 기동 |

각 서버노드의 정보와 이벤트를 전달받고, 이를 바탕으로 서버노드의 각 서비스별 프로세스, probe, 그리고 NM의 상태를 나타내는 서비스 상태 테이블을 유지 관리한다. 또한, 각 서버노드의 클러스터 네트워크와 서비스 네트워크를 통해 ping 메시지를 전송하여, 네트워크의 고장여부를 판단한다. 서비스 상태 테이블은 시스템을 모니터하는 시스템 probe와 DB, 리눅스 가상 서버, 노드의 네트워크 상태와 NM의 상태를 활성과 비활성으로 나타낸다. 표 2는 CM이 발생시키는 이벤트와 그에 따른 수행절차를 나타낸다.

CM은 네트워크의 고장여부와 서비스 상태 테이블, 그리고 전달받은 이벤트를 바탕으로 클러스터 시스템내의 모든 자원과 서비스를 통합 관리하여 이벤트를 발생시키며, 전달받은 이벤트를 해석하여 적절한 서비스 handler에게 이벤트를 전달하여 그에 따른 절차를 수행하게 한다. 한편 CM이 실행되는 관리노드에 오류가 발생하면, 전체 클러스터 기반 DBMS의 오류로 나타나기 때문에, 이를 방지하기 위하여 백업관리노드를 둔다. 백업관리노드에는 백업 CM이 실행되며, 관리노드의 CM과의 통신을 통해 CM이 유지하는 모든 정보를 정해진 시간간격으로 전달받아 유지한다. 일단 CM의 관리노드에 CM의 오류 및 네트워크의 고장이 발생하면, 이를 감지한 백업관리노드의 BCM(Backup Cluster Manager)은 자신의 노드를 관리노드로 바꾸고 CM의 역할을 수행한다. 또한, 서비스 가능한 서버노드 가운데 하나를 자신의 백업관리노드로 선택한다. 한편 BCM의 오류나 백업관리노드의 오류는 관리노드의 CM이 감지하며, CM이 오류를 감지하면 서비스 가능한 서버노드 가운데 하나를 자신의 새로운 백업관리노드로 선택한다.

### 3.3 고장의 종류 및 그에 따른 복구 절차

클러스터 시스템을 구성하는 각 노드는 관리노드, 백업관리노드, 데이타베이스 서버의 역할을 수행한다. 따라서 클러스터 시스템의 고장은 관리노드의 고장, 백업관리노드의 고장, 데이타베이스 서버노드의 고장으로 구분할 수 있다. 또한 클러스터 관리기 내의 네트워크는 각 노드들 간의 통신을 수행하는 클러스터 네트워크와, 사용자의 서비스 요청을 받아들이고 처리된 결과를 사용자에게 전달하는 서비스 네트워크로 구성되므로, 네트워크의 고장여부에 따라 위의 세 가지 고장에 따른 복구절차를 수행하여야 한다. 표 3은 네트워크 상태에 따른 고장의 종류를 나타낸다.

표 3 네트워크 상태에 따른 고장의 종류

| 네트워크 종류 / 노드상태 | 클러스터 네트워크 | 서비스 네트워크 |
|---|---|---|
| 정상 | O | O |
| 서비스네트워크 단절 | O | X |
| 클러스터네트워크 단절 | X | O |
| 노드완전고장 | X | X |

첫째, 서비스 네트워크 단절은 각 노드들 간의 통신은 가능하지만 사용자로부터 서비스 요청을 받아들일 수 없거나, 처리된 결과를 사용자에게 전달할 수 없는 상태이다. 둘째, 클러스터 네트워크 단절은 사용자의 서비스 요청을 각 노드로 재분배 할 수 없는 상태이거나, 노드간의 통신이 이루어질 수 없는 상태이다. 마지막으로, 노드 완전고장은 클러스터 네트워크 및 서비스 네트워크가 단절되어 해당 노드가 아무런 기능을 할 수 없는 상태이다.

#### 3.3.1 관리노드의 고장과 그에 따른 복구절차

클러스터 시스템에서 관리노드는 사용자의 서비스 요청을 재분배하고 전체 클러스터 기반 DBMS를 관리하

기 때문에 관리노드의 고장은 전체 시스템의 고장으로 연결된다. 먼저 관리노드의 서비스 네트워크 단절의 경우, 사용자로부터의 서비스 요청을 관리노드가 받을 수 없기 때문에 클러스터 시스템의 전체 고장으로 여겨진다. 아울러, 관리노드의 클러스터 네트워크가 단절되면, 사용자의 서비스 요청을 다른 서버노드로 재분배 할 수 없기 때문에, 관리노드를 제외한 모든 서버노드의 고장으로 여긴다. 마지막으로, 노드완전고장은 위 두 가지 고장의 경우를 합한 결과를 발생시킨다. 이를 방지하기 위해 관리노드의 고장을 감지하고, 관리노드 고장 발생시 관리노드의 역할을 대행할 백업관리노드를 둔다. 표 4는 관리노드의 고장과 그에 따른 복구절차를 나타낸다.

먼저 관리노드는 자신을 제외한 서버노드로 ping 메시지를 보내고 모든 서버노드로부터 ping 메시지에 대한 응답을 받지 못했을 경우, 자신의 네트워크 단절로 간주하고 관리노드를 종료한다. 백업관리노드는 관리노드와의 통신에 대한 응답을 받지 못하거나 관리노드로 전송한 ping 메시지에 대한 응답이 없을 경우, 이를 관리노드의 고장으로 간주하고 관리노드의 역할을 수행한다. 이때 그림 5와 같은 모호한 상황이 발생할 수 있다.

즉, 그림 5의 (a)는 관리노드를 제외한 모든 서버노드의 클러스터 네트워크가 단절된 상황이며, 그림 5의 (b)는 관리노드의 클러스터 네트워크가 단절된 상황이다. 하지만 두 가지 상황 모두 ping 메시지에 대한 결과가 자신을 제외한 모든 서버노드의 클러스터 네트워크 단절로 나타나 두 가지 상황을 구분할 수 없게 된다. 이는 서비스 네트워크의 단절에서도 똑같은 상황이 발생한다. 따라서 이러한 모호한 상황을 구분하기 위하여 ping 메시지에 대한 결과를 항상 백업한다. 즉, 그림 5와 같은 상황이 발생하면 과거의 결과와 현재의 결과를 비교하여, 둘이상의 서버노드의 네트워크가 동시에 단절되었을 경우를(실제로, 동시에 단절되는 것이 거의 불가능하다고 간주하기 때문에) 관리노드의 네트워크 단절로 간주하고, 표 4에 나타난 복구절차를 수행한다.

3.3.2 백업관리노드의 고장과 그에 따른 복구절차

클러스터 시스템에서 백업관리노드는 관리노드와의 통신을 통해 관리노드가 유지하고 있는 모든 정보를 주기적으로 백업받고 관리노드를 모니터한다. 백업관리노드의 고장 또한 앞서 언급한 관리노드의 고장과 같은 모호한 상황이 발생한다. 백업관리노드의 이러한 모호한 상황은 ping 결과를 백업하여 과거와 현재의 ping 메시지에 대한 상태를 비교한다. 이때, 둘 이상의 노드의 네

표 4 관리노드의 고장과 그에 따른 복구절차

| 서버종류 / 노드상태 | 관리노드 | 백업관리노드 |
|---|---|---|
| 서비스 네트워크 단절 | · ifconfig eth1:0 VIP down<br>· ipvsadm -C<br>· 수행중인 DB, Sysmon 서비스를 종료한다.<br>· NM을 종료한다.<br>· CM을 종료한다. | · ifconfig eth1:0 VIP up<br>· 서비스 가능한 서버노드를 부하분산대상에 추가한다.<br>· 관리노드역할을 수행한다.<br>· 서비스 가능한 서버노드중의 하나를 백업관리모드로 정하고 Backup CM을 실행시킨다. |
| 클러스터 네트워크 단절 | | |
| 노드 완전고장 | | |

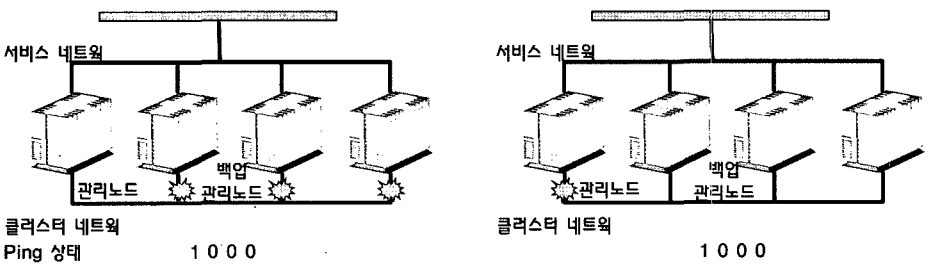



그림 5 관리노드 고장 구분의 모호한 상황

표 5 백업관리노드의 고장과 그에 따른 복구절차

| 서버종류 / 노드상태 | 관리노드 | 백업관리노드 |
|---|---|---|
| 서비스 네트워크 단절 | · 백업관리노드를 부하분산대상에서 제거한다.<br>· 서비스 가능한 서버노드 중의 하나를 백업관리모드로 정하고 Backup CM을 실행시킨다. | · 수행중인 DB, Sysmon 서비스를 종료한다.<br>· NM을 종료한다.<br>· Backup CM을 종료한다. |
| 클러스터 네트워크 단절 | | |
| 노드 완전고장 | | |

트워크가 동시에 고장이 발생하였을 경우, 백업관리노드의 네트워크 단절로 간주하고 백업관리노드의 역할을 종료하며, 관리노드가 서비스 가능한 다른 노드를 백업관리노드로 선택한다.

3.3.3 DB 서버노드의 고장과 그에 따른 복구절차

클러스터 시스템 환경에서 클러스터 기반 DBMS는 디스크를 공유하기 때문에 한 서버노드의 잘못된 동작은 전체 클러스터 기반 DBMS에 영향을 미칠 수 있다. 따라서 서버노드가 잘못된 동작을 하여 전체 클러스터 기반 DBMS의 데이타 무결성에 영향을 미치지 않도록 네트워크 고장 종류에 따른 복구 절차를 수행한다. 첫째, 서비스 네트워크가 단절될 때 사용자의 서비스 요청에 대한 처리결과를 사용자에게 전송할 수 없으므로, 해당 노드는 현재 수행중인 트랜잭션을 중단하고 완료가 안된 트랙잭션의 복구절차를 수행한다. 또한 DBMS가 사용자의 서비스 처리 요구를 받지 않도록, 리눅스 가상서버의 부하분산대상에서 제거한다. 둘째, 클러스터 네트워크가 단절될 때, 해당 서버노드는 관리노드로부터 사용자의 서비스 요청을 전달받을 수 없으며, 다른 서버노드에 데이타를 전달할 수 없으므로 실행중인 트랜잭션의 복구절차를 수행하고 종료한다. 마지막으로, 노드의 완전 고장 상태가 발생하면 해당 서버노드를 부하분산 대상에서 제거하고, 다른 서버노드가 이를 알 수 있도록 하고 아울러 복구 절차를 수행한다.

## 4. 클러스터 관리기의 구현 및 성능고찰

본 절에서는 클러스터 기반 DBMS인 iBASE/ Cluster를 이용한 클러스터 관리기의 구현 및 성능고찰을 수행한다. 이를 위한 클러스터 관리기의 구현환경은 Intel Pentium-III 450 CPU, 128 MByte의 메모리의 4대의 PC로 구성하며, Redhat 7.1을 사용하고 커널 버전은 2.4.5을 사용한다. 컴파일러는 gcc 2.96 및 Make ver 3.79.1를 사용하였고, 리눅스 가상 서버로 0.81버전을 사용한다.

### 4.1 클러스터 관리기의 테스팅

본 논문에서 구현된 클러스터 관리기를 iBASE/ Cluster를 이용하여 정상동작여부를 검사한다. 구현 및 테스팅에 사용된 클러스터 시스템은 4대의 서버로 구성되며, 각 서버에 iBASE/Cluster DBMS가 설치되어 동작한다. 테스팅은 정상동작상태, 관리노드에 오류가 발생했을 때, 백업관리노드에 오류가 발생했을 때, iBASE/ Cluster 서버 노드에 오류가 발생했을 때, 그리고 두 노드가 연속적으로 오류가 발생하는 다중 오류 상황으로 구분하여 각각의 오류상황에 다른 복구절차가 정상적으로 실행되는 지를 검사한다.

첫째, 정상동작상태를 테스팅하기 위하여 첫 번째 서버노드는 관리노드의 역할을, 두 번째 서버노드는 백업관리노드의 역할을 수행하도록 한다. 사용자의 서비스 요청은 관리노드인 첫 번째 서버로 전송되며 관리노드의 리눅스 가상 서버는 현재 서비스 가능한 4개의 서버노드 가운데 한 서버노드로 round-robin 알고리즘에 의해 순서대로 서비스 요청을 재분배한다. 그림 6은 정상동작상태일 때의 클러스터 관리기의 사용자 인터페이스를 나타낸다. 현재 클러스터 시스템을 구성하는 4개의 서버노드가 정상동작중이며, 그중 첫 번째 서버노드는 관리노드의 역할을 수행하며, 두 번째 서버노드는 백업관리노드의 역할을 수행한다. 또한 클러스터 관리기의 각 컴포넌트 상태에서 현재 모든 서버노드의 데몬프로그램(sys로 표시) 및 iBASE/Cluster가(iBase로 표시) 정상동작중이며, CM과의 통신을 관리하는 NM이 정상동작하며 모든 서버노드가 서비스중임을 알 수 있다. 현재 리눅스 가상서버는 관리노드인 첫 번째 서버노드에서 정상동작중이며, 클러스터 기반 DBMS의 전역잠금

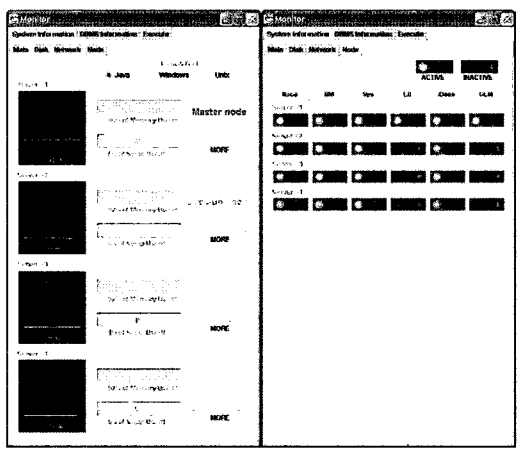



그림 6 정상동작 상태에서 사용자 인터페이스

표 6 노드 상태에 따른 DB 서버의 복구절차

| 서버종류 / 노드상태 | 정상 서버노드 | 고장 발생 서버노드 |
|---|---|---|
| 서비스 네트워크 단절 | 해당 사항 없음 | · 부하분산대상에서 제거<br>· 수행중인 트랜잭션 복구<br>· 데이타베이스 종료 |
| 클러스터 네트워크 단절 | · 고장이 발생한 서버의 데이타 복구를 위해 회복절차 수행 | |
| 노드 완전고장 | | |

관리기인 GLM(Global Lock Manager)이 마스터 노드인 첫 번째 노드에서 실행중이다.

둘째, 노드의 종류에 따른 단일 노드의 오류 발생 시 클러스터 시스템의 동작 여부를 검사한다. 먼저 관리노드의 오류를 검사하기 위해, 관리노드의 클러스터 네트워크 케이블의 연결을 차단하여 관리노드를 노드 격리 상태로 만든다. 그림 7은 관리노드에 오류가 발생 시 클러스터 관리기의 사용자 인터페이스를 나타낸다. 즉, 관리노드의 역할이 백업관리노드로 동작하던 두번째 관리노드로 바뀌었으며, 서비스 가능한 다른 서버노드중의 한 서버노드가 새로이 백업관리노드로 동작한다. 또한 클러스터 관리기의 각 컴포넌트 상태에서 관리노드의 NM, Sys, LB, iBase가 INACTIVE 상태임을 알 수 있고, 첫 번째 서버노드에서 실행되던 리눅스 가상 서버와 iBASE/Cluster의 GLM이 현재 관리노드인 두 번째 서버노드에서 정상적으로 동작한다. 이번에는 백업관리노드의 오류발생시 백업관리노드가 다른 서버노드로 역할이 바뀌는 지를 검사한다. 이를 위해 백업관리노드의 클러스터 네트워크 케이블의 연결을 차단하여 백업관리노드를 노드격리 상태로 만든다. 테스팅 결과는 그림 9의 형태로 나타난다. 즉, 그림 8에서 백업관리노드로 동작중인 두 번째 서버가 서비스 불가능한 상태로 되며, 다른 서버노드 중 한 노드가 백업관리노드로 선택되어 동작한다. 마지막으로, 데이타베이스가 실행되는 세 번째 서버노드에서 데이타베이스에 오류가 발생할 때를 테스팅한다. 이를 위해 세번째 서버노드의 클러스터 네트워크 케이블의 연결을 차단하여 노드격리 상태로 만든다. 테스팅 결과는 네트워크 연결이 차단된 세 번째 서버의 상태가 서비스 불가능한 상태로 나타나며(그림 7), 이를 제외한 나머지 노드들은 정상적으로 동작한다.

마지막으로, 관리노드, 백업관리노드, 데이타베이스 서버노드에서 오류가 발생한 뒤에 또 다른 종류의 노드

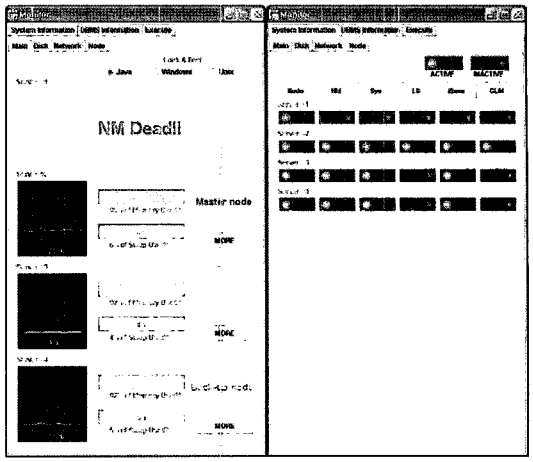



그림 7 관리노드의 오류발생시 사용자 인터페이스

오류가 연속되는 다중 오류 경우를 테스팅한다. 이를 위해 먼저 관리노드, 백업관리노드, 데이타베이스 서버노드에서 오류가 발생한 뒤의 클러스터 시스템 내의 상태를 표 7에 요약한다. 정상상태에서 노드 종류에 따른 오류 발생 후에, 오류 발생 노드를 제외한 노드들이 정상적으로 서비스를 수행한다. 즉, 3개의 노드로 이루어진 클러스터 시스템과 같은 상태가 된다. 따라서 다중 오류 발생 상황은 단일노드의 오류발생시 복구 절차를 3개의 노드로 이루어진 클러스터 시스템에서 테스팅하는 것과 같은 결과를 보인다.

### 4.2 클러스터 관리기의 성능 고찰

본 절에서는 iBASE/Cluster를 이용한 클러스터 관리기의 성능을 분석한다. SCMS는 클러스터 시스템을 위한 관리기로서 오류 발생 상황에 대한 DBMS의 장애 극복 절차를 지원하지 않으며, OCMS는 Oracle이라는 특정 DBMS를 위한 클러스터 관리기로서 iBASE/

표 7 노드 오류발생 전과 후의 시스템 상태
(M: 관리노드, B: 백업관리노드, DB: 데이타베이스 서버노드)

| 오류발생 전 | | | | 오류발생노드 | 오류발생 후 | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 관리노드 | 오류 | 정상 | 정상 | 정상 |
| | | | | | | M | DB | B |
| 정상 | 정상 | 정상 | 정상 | 백업관리노드 | 정상 | 오류 | 정상 | 정상 |
| | | | | | M | | DB | B |
| M | B | DB | DB | 데이타베이스 서버노드 | 정상 | 정상 | 오류 | 정상 |
| | | | | | M | B | | DB |

Cluster의 장애 극복 절차를 지원하지 않아, 이들과의 직접적인 성능 비교가 불가능하다. 따라서 본 논문에서는 구현된 클러스터 관리기의 관리노드, 백업관리노드, 데이타베이스 서버노드의 고장을 감지하고, 각각의 고장에 따른 복구 절차를 수행하는데 소요되는 시간을 측정하여 성능고찰을 수행한다. 표 8은 각각의 고장 감지와 이에 따른 복구절차 수행에 소요되는 시간을 나타낸다.

표 8 평균 고장 감지 시간 및 이에 따른 평균 복구절차 수행시간(단위:초)

| | 고장 감지 시간 | 복구절차 수행시간 |
|---|---|---|
| 관리노드의 고장발생시 | 0.91 | 0.78 |
| 백업관리노드의 고장발생시 | 0.89 | 0.51 |
| 데이타베이스 서버노드의 고장발생시 | 0.81 | 0.71 |

첫째, 관리노드의 고장이 발생하면 이를 모니터하는 백업관리노드가 네트워크 상태와 관리노드로 ping 메시지를 보내고 이에 대한 응답을 통해 관리노드의 고장을 감지하고, 자신이 관리노드의 역할을 수행한다. 여기서 ping 메시지에 대한 응답은 최소주기를 2초로 설정하였다. 표에 나타난바와 같이 고장을 감지하는데 소요되는 시간은 0.91초이며, 백업관리노드에서 관리노드로 전환되는데 소요되는 시간은 0.78초가 소요된다. 이를 통해 고장 감지부터 복구 절차 수행까지 평균 1.69초가 소요되어 최소주기인 2초 내로 수행하므로 효율적이라 할 수 있다. 아울러 백업관리노드에서 관리노드로 전환되는 0.78초 동안, 백업관리노드는 rkr상 IP를 설정하고, 리눅스 가상 서버의 부하분산 대상의 서버 노드를 설정한다.

둘째, 백업관리노드의 고장이 발생하면 이를 관리노드가 감지하고 서비스 가능한 다른 노드를 백업관리노드로 선택한다. 백업관리노드의 고장을 감지하는데 소요되는 시간은 0.89초이며, 백업관리노드의 역할 수행을 위한 복구절차는 0.51초가 소요된다. 여기서 복구절차 수행 시간이 0.51초로 다른 노드의 복구 절차 수행시간보다 빠른 이유는, 백업관리노드로 선택된 노드는 단순히 관리노드와 네트워크 연결만을 설정하여, 이후 관리노드의 서비스 상태 테이블을 주기적으로 백업받기 위한 절차만 준비하면 되기 때문이다.

마지막으로, 데이타베이스 서버노드의 고장이 발생하면 관리노드가 이를 감지하고 이에 따른 복구절차를 수행한다. 데이타베이스 서버노드의 고장을 감지하는데 소요되는 시간은 0.87초이며, 복구절차를 수행하는데 소요되는 시간은 0.71초이다. 이를 통해 고장 감지부터 복구절차 수행까지 평균 1.52초가 소요되어 최소주기인 2초 내로 수행하므로 효율적이라 할 수 있다. 한편 복구절차 수행 시간이 0.71초로 백업관리노드의 복구 절차 수행시간보다 느린 이유는, 데이타베이스 서버노드의 고장이 감지되면 해당노드에서 데이타베이스 인스턴스를 강제 종료시켜 데이타베이스 인스턴스가 점유하고 있는 자원들을 강제 반환하도록 하며, 또한 해당 노드로 사용자의 서비스 요청이 전달되지 않도록 관리노드가 이를 부하분산대상에서 제거하는 작업이 요구되기 때문이다.

아울러 본 논문에서 구현한 클러스터 관리기의 오류감지 및 복구수행시간의 효율성을 제시하기 위해 OCMS를 포함하는 Oracle 9i RAC 버전의 매뉴얼을 검토하였으며, 그 결과 Oracle 9i OCMS의 경우 Oracle DBMS 서버 노드의 고장을 감지하고 복구하는데 최소 20초 이상의 시간이 소요되어[12], 본 논문에서 구현한 클러스터 관리기가 오류감지 및 복구수행시간 측면에서 (2초 이내) 매우 성능이 뛰어남을 알 수 있다. 또한 본 논문에서 구현한 클러스터 관리기와 OCMS 를 비교하면 표 9와 같다. 첫째, OMCS의 경우 cluster manager는 각 서버노드에 존재하며 다른 서버노드의 cluster manager와 통신을 통해 전체 시스템을 관리한다. 만약 N 개의 서버노드가 존재한다면, OCMS는 시스템을 관리하기 위해 (N-1)*N 개의 메시지를 교환해야 한다. 하지만 제안하는 클러스터 관리기는 단지 2N 개의 메시지의 교환을 필요로 한다. 따라서, 제안하는 클러스터 관리기가 OCMS 보다 네트워크 자원을 효율적으로 사용할 수 있음을 알 수 있다. 둘째, OCMS는 다른 서버노드의 오류감지를 위해 공유 디스크상의 quorum 파티션을 사용한다. 따라서 서버의 수가 증가할수록 quorum 파티션에 접근하고자 하는 횟수가 증가하여 성능저하를 초래한다. 하지만 제안하는 클러스터 관리기는 과거 상태 정보를 바탕으로 오류를 감지한다. 즉, 2개 이상의 서버노드가 동시에 오류가 발생할 수 없다는 가정하에 2개이상의 노드가 동시에 오류가 발생하여 네트워크 통신이 불가능할 경우, 현재의 상태 정보와 과거의 상태정보 비교를 통해 현재 노드에 오류가 발생한 것을 감지한다. 논 논문에서 제안하는 과거 상태 정보를 통해 오류를 감지하는 방법은 서버 노드의 수에 영향을 받지 않아서 확장성 측면에서 OCMS 보다 효율적이다. 마지막으로, OCMS는 공유 디스크의 오류를 감지하여 DBMS 인스턴스들이 공유디스크에 접근하지 못하도록

표 9 제안하는 클러스터 관리기와 OCMS의 비교

| | 제안하는 클러스터 관리기 | Oracle OCMS |
|---|---|---|
| 서버간 네트워크 통신 | 중앙집중형 | 분산형 |
| 오류 감지 | 과거 상태 정보 | Quorum 파티션 |
| 공유 디스크 오류 감지 | 미지원 | 지원 |

복구절차를 수행한다. 하지만 제안하는 클러스터 관리기는 현재 공유 디스크 오류를 감지하지 못하는 단점이 있다.

## 5. 결론 및 향후 연구

최근 인터넷 환경에서 요구되는 24시간 무정지 서비스 요구를 효과적으로 처리하기 위해, 여러 대의 단일 서버를 고속의 네트워크로 연결한 클러스터 기반 DBMS의 연구가 국내외적으로 활발히 진행중이다. 그러나 이러한 클러스터 기반 DBMS를 효율적으로 관리하고, 아울러 운영중인 컴퓨터가 고장 등의 이유로 운용할 수 없을 때에도 전체 클러스터 기반 DBMS가 정상적인 동작을 할 수 있도록 지원하는 관리 도구에 대한 연구는 미흡한 실정이다. 본 논문에서는 클러스터 기반 DBMS의 효율적인 관리를 위해 고가용성 클러스터 관리기를 설계하고 구현하였다. 이는 클러스터 시스템을 이루는 각 서버노드에서 시스템의 주요 자원을 모니터하고, 이를 통해 시스템, 네트워크, 데이타베이스의 오류를 감지하고, 오류 발생 시 복구절차를 수행한다. 따라서 전체 클러스터 기반 DBMS가 오류에 상관없이 계속적으로 정상적인 서비스를 수행할 수 있도록 클러스터 시스템을 관리한다. 또한 클러스터 관리기를 통해 사용자가 클러스터 시스템을 구성하는 각 노드들의 자원 활용 상태를 보다 쉽게 파악하고 클러스터 시스템을 관리할 수 있도록 하였다. 아울러 클러스터 관리기를 클러스터 기반 DBMS인 iBASE/Cluster를 이용하여 구현하였고, 관리노드, 백업관리노드, 데이타베이스 서버노드 그리고 다중노드의 오류 상황에 대해 테스팅하였다. 한편 오류 발생시 그에 따른 복구 절차를 수행함으로써 오류 발생 후에도, 전체 클러스터 기반 DBMS가 정상적인 서비스를 수행할 수 있도록 하였다.

향후 연구로는 첫째, 공유 디스크에 대한 오류 감지 및 오류 발생시 복구절차를 수행할 수 있도록 개선하고, 둘째, 각 노드에서 모니터한 부하 및 자원 활용 상태에 기반한 효율적인 부하분산 알고리즘을 설계하여, 사용자의 서비스 요청을 보다 효율적으로 분산시켜 클러스터 기반 DBMS의 전체적인 성능향상을 도모하는 것이다.

## 참 고 문 헌


[1] 김진미, 온기원, 김학영, 지동해, "클러스터링 컴퓨팅 기술", 1999.

[2] R. Buyya, High Performance Cluster Computing Vol 1&2, Prentice Hall, 1999.

[3] "High Performance Communication," http://www-csag.cs.uiuc.edu/projects/communication.html.

[4] 유찬수, "리눅스 클러스터링", 정보과학회 논문지, 제18권 제2호 2000년, pp. 33-39.

[5] 최재영, 황석찬, "클러스터를 위한 소프트웨어 도구", 정보과학회지, 제18권 3호, pp. 40-47.

[6] Gregory, F.Pfister, In Search of Clusters 2nd Edition, Prentcs-Hall, 1998.

[7] Linux Clustering, http://dpnm.postech.ac.kr/cluster/index.htm

[8] Oracle Corporation, "Oracle 8i Administrator's Reference Release3(8.1.7) for Linux Intel," chapter 7, Oracle Cluster Management Software, 2000.

[9] P. Uthayopas, J. Maneesilp, P. Ingongnam, "SCMS: An Integrated Cluster Management Tool for Beowulf Cluster System," Proc. of the Int'l Conf. on Parallel and Distributed Techniques and Applications, pp. 26-28 June 2000.

[10] 리눅스 가상 서버, http://www.linuxvirtualserver.org

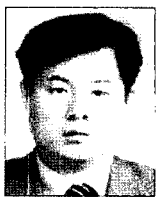
[11] 김홍연, 진기성, 김준, 김명준, "iBASE/Cluster: 클러스터 환경을 위한 바다-IV의 확장", 한국정보처리학회 추계학술발표대회 논문집, 제9권 제2호, 2002.

[12] Oracle Corporation, "Oracle 9i Administrator's Reference Release2(9.2.0.1.0)," chapter F, Oracle Cluster Management Software, 2002.



김 영 창

2001년 전북대학교 컴퓨터공학과(공학사). 2003년 전북대학교 컴퓨터공학과(공학석사). 2004년~현재 전북대학교 컴퓨터공학과(박사재학). 관심분야는 공간 네트워크 데이타베이스, 클러스터 컴퓨팅


장 재 우

1984년 서울대학교 전자계산기공학과(공학사). 1986년 한국과학기술원 전산학과(공학석사). 1991년 한국과학기술원 전산학과(공학박사). 1996년~1997년 Univ. of Minnesota, Visiting Scholar. 2003년~2004년 Penn State Univ., Visiting Scholar. 1991년~현재 전북대학교 컴퓨터공학과 교수. 관심분야는 공간 네트워크 데이타베이스, 상황인식, 하부저장구조



김 홍 연

1992년 8월 인하대학교 통계학과(이학사). 1994년 8월 인하대학교 전자계산학과(공학석사). 1999년 8월 인하대학교 전자계산학과(공학박사). 1999년 8월~현재 한국전자통신연구원 선임연구원. 관심분야는 네트워크 스토리지 및 파일시스템, 클러스터 컴퓨팅, DBMS