클러스터 인식 응용 프로그램의 구현 기법

김영수* · 조익성* · 임재홍*

*한국해양대학교 전자통신공학과

An Implementation Methodology of the Cluster Aware Application Program

Yeong-Su Kim* · Ik-Sung Cho* · Jae-Hong Yim**

*Dept. of Electronics & Communication Eng, Korea Maritime University

E-mail : k971064g@hanbada.kmaritime.ac.kr

요 약

본 논문은 클러스터 환경에서 자동적인 작업인계, 빠른 에러 복구, 관리의 편의성을 제공하는 클러스터 인식 응용 프로그램의 구현기법에 관하여 논한다. 이를 위하여 클러스터의 전체적인 시스템 구조 및 알고리즘을 제시한다.

본 논문의 타당성 검토를 위하여 소켓 응용 프로그램과 이 응용 프로그램에 대한 클러스터 리소스 DLL, 관리 DLL을 설계 구현하고, 마이크로소프트 클러스터 서버를 이용하여 응용 프로그램을 한 서버에서 나머지 서버로 재배치함으로써 서버내의 응용프로그램이 결함이 생긴 서비스에 영향을 받지 않고 계속적으로 동작한다는 결론을 얻을 수 있었다.

ABSTRACT

This paper describes the development of cluster-aware application that provides advanced services such as automatic failover, faster error recovery, easy administration and scalability in clustering environment. For These, total system architecture and algorithm are proposed.

For the verification this paper, socket application and cluster resource DLL, administration DLL for the application are implemented and tested. Using the microsoft cluster server, The individual failed services was relocated from one server to another. The result shows that allows applications on the original server to continue running, unaffected by the failed service.

Ⅰ. 서 론

시스템의 성능이 높아지고 기능이 다양해질수록 그 동작이 복잡해지고 장애(fault)의 가능성이 높아지게 된다. 높은 가용성(availability)을 이룰 수 있는 한 가지 전통적인 하드웨어 구조는 완전히 똑같은 컴포넌트를 가진 시스템을 중복시키는 것이다. 예전의 컴퓨팅 환경, 즉 메인프레임을 이용하여 연산과 저장, 출력 등을 모두 처리하는 구성에서는 무정지 시스템을 구현하여 장애에 대비하였다[1]. 또한 이러한 하드웨어를 활용하기 위한 전통적인 소프트웨어 모델은 일차 시스템이 응용 프로그램을 실행하고 있는 반면, 다른 시스템은 일차 시스템의 에러가 발생하는 경우에 작업을 넘겨 받기 위하여 대기상태로 있는 것이다. 이 방식의 단점은 시스템의 산출에는 전혀 도움이 안되면서도 하드웨어 비용이 증가하며, 간헐적인 응용 프로그램의 에러에 대비할 수 없다는 것이다.

그러나 이제는 보편적으로 컴퓨팅 환경과 사용자의 마인드가 중앙집중식 환경에서 분산처리 환경 또는 클라이언트/서버 환경으로 변화되었다. 이에 따라 새로운 환경에서 장애에 대비할 새로운 방법이 필요하게 되었고, 그것이 바로 클러스

터라는 개념이다.

분산처리 환경에서는 작업에 따라 서버를 구분하여 처리 성능을 개선하도록 하는 클라이언트/서버 구현이 필연적이다. 이것은 예전에 메인프레임 혼자서 다 처리하던 연산, 저장, 출력, 통신 등과 같은 작업들을 여러 대의 서버가 각각 성능에 따라 하나 또는 그 이상씩 나누어서 전담하도록 하고 클라이언트에서는 사용자 인터페이스를 구동하여 서비스를 제공하는 서버를 찾아 그 작업을 해주도록 요청하는 방법이다[2]. 클러스터는 이러한 환경에서 서버의 서비스가 중단되는 시간을 최소화하는 기법을 말한다.

구체적으로 서버 시스템 및 네트워크, 저장장치 등의 다중화를 통하여 장애에 대비하는 하드웨어적인 준비가 필요하며 소프트웨어적으로는 복수의 서버를 기억하고 있다가 서비스가 가능한 서버를 찾아 요청을 보내도록 클라이언트 프로그램을 작성하거나 미들웨어(middleware)를 사용하기도 한다. 컴퓨팅 응용 프로그램에서 가용성과 확장성을 제공하는 구조는 클러스터이다. 클러스터링은 두 서버의 작업 부담이 자동적으로 다른 서버로 작업인계(failover)할 수 있게 하여, 고도의 가용성을 갖춘 분산 컴퓨팅 환경을 만드는 것이다. 또한 관리자는 작업 부하의 균형을 유지하기 위하여 수작업으로 작업인계를 시작하거나, 작업의 중단 없이 계획된 유지보수를 위하여 서버를 로드(load)할 수 있다.

본 논문에서는 이러한 자동적인 작업인계 기능, 신속한 복구, 쉬운 관리의 편의성과 높은 확장성을 위하여 클러스터를 충분히 활용하는 클러스터 인식 응용 프로그램의 구현에 관하여 논한다. 이를 위하여 클러스터 서버의 구조 및 알고리즘을 제시하고, 실제로 2대의 윈도우즈 NT 서버로 구성된 마이크로소프트 클러스터 서버 환경에서 하나의 서버가 다운되는 경우, 에러가 발생한 시스템의 작업을 나머지 서버가 복구하여 클러스터 내의 다른 시스템에 분산시키는 클러스터 인식 응용 프로그램의 구현기법에 관하여 논한다.

본 논문의 2장에서는 고장허용(fault tolerance) 기법과 마이크로소프트 클러스터의 구조에 대하여, 3장에서는 클러스터 인식 응용 프로그래밍 기법에 대하여 기술하고, 4장에서는 실험 결과 및 고찰에 관하여, 마지막으로 5장에서는 결론을 서술한다.

II. 고장허용 기법

1. 무정지 시스템

데이터베이스 서비스, 파일 서비스, 프린트 서비스 등을 제공하는 서버는 항상 동작 상태에 있어야 하므로 주 서버와 백업 서버로 서버 시스템을 이중화 하여, 그림 1과 같이 주 서버에서 장애가 발생할 경우 그림 2와 같이 백업 서버가 주 서버에서 운영중이던 프로세스를 인계받아 서버의 역할을 대신 수행하도록 하는 기법이다. 이 방식은 메인 시스템과 이 시스템이 정지했을 때를 대비하여 항시 대기상태로 있는 백업 시스템으로 구성되며, 두 시스템은 항상 모든 동작 상태를 일치시켜 메인 시스템 장애시 백업 시스템이 계속 동작을 이어가도록 하는 방법이다. 그러나 위와 같은 예에서는 자원의 낭비가 크다는 단점이 있기 때문에 평상시에도 백업시스템을 활용하는 방법을 생각할 수 있다.

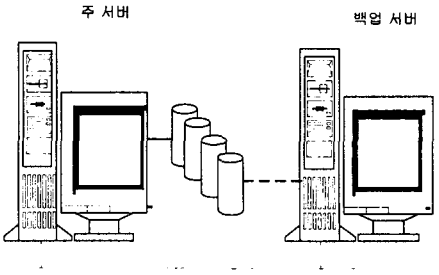

그림 1. 결함 전의 무정지 시스템
Fig 1. Non-stop system before failure

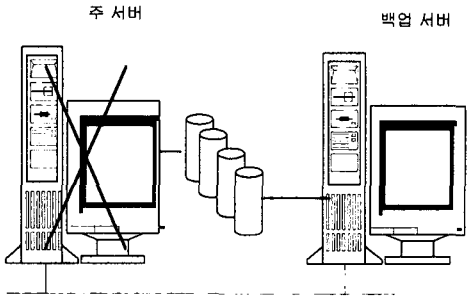

그림 2. 결함 후의 무정지 시스템
Fig 2. Non-stop system after failure

2. 클러스터

클러스터란 2개 이상의 노드를 느슨하게 연결된(loosely coupled) 시스템으로 구축한 것으로, 컴퓨팅 차원에서 고도의 가용성과 확장성 및 유연성, 관리의 편의성을 제공하는 솔루션이다. 클러스터가 제공하는 가용성 수준은 무정지 시스템과 비교되는데, 클러스터는 한 서버의 완벽하게 미러(mirror)된 백업을 필요로 하지 않기 때문에

비용면에서 매우 효과적이다.

넓은 의미에서, 클러스터는 단일 시스템처럼 서로 작동하는 독립 시스템의 그룹이다. 클라이언트는 클러스터가 단일 서버인 것처럼 상호작용하며, 클러스터의 구성은 가용성, 관리의 편의성과 확장성을 제공하기 위하여 사용된다. 두 시스템을 SCSI(Small Computer System Interface)버스를 공유하는 방식으로 결합하여 클러스터 또는 단일 시스템 환경을 만든다. 클라이언트는 클러스터 내의 어떤 서버를 사용하는지 알 필요없이 클러스터의 모든 자원(디스크, 파일, 데이터 베이스)에 접근 할 수 있다. 그림 3은 클러스터의 전체적인 구성을 나타낸다.

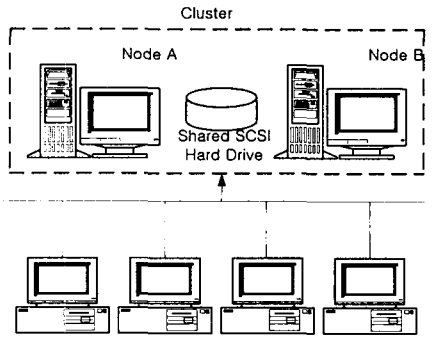

그림 3. 클러스터

Fig 3. Cluster

3. 작업인계 객체와 그룹

가. 작업 인계(fail-over)

클러스터는 여러 종류의 작업인계 객체를 지원하는데 데이터베이스 응용 프로그램과 같은 클러스터 서비스와 네트워크 공유자원과 같은 클러스터 리소스가 그것이다. 주 서버에 이상이 발생하면, 클러스터 소프트웨어는 자동적으로 나머지 서버를 통한 백업 패스로 리소스를 마치 아무런 이상도 발생하지 않았던 것처럼 제공하여 준다. 이렇게 작업인계 객체를 투명하게 재할당 해주는 것을 작업 인계라고 하며, 이러한 작업인계 객체에는 프린터나 TCP/IP 어드레스, 파일 공유 리소스, 사용자 정의 객체 등이 있다.

클러스터링 기능이 없는 서버 응용 프로그램이 클러스터링을 갖추게 되면 응용 프로그램의 자동적인 작업 인계 기능, 또한 신속한 복구, 쉬운 관리 편의성과 높은 확장성을 제공한다. 위와 같은 기능을 수행하기 위해서는 클러스터는 응용 프로그램을 인식해야만 한다[3][4].

나. 페일백(fail-back)

이것은 이상이 발생한 서버가 다시 운용이 가능해졌을 때 클러스터에 어떤 일이 일어나는 가에 따라 설명될 수 있다. 즉 페일백이 가능한 상태가 되어 있다면 작업인계 그룹은 자동적으로 원래 수행되던 서버로 가서 수행이 되고, 페일백이 가능한 상태가 되어 있지 않다면 작업인계 그룹은 그대로 2차 서버에서 수행된다. 페일백 기능은 클러스터 멤버의 작업 분산(load balancing)을 위한 중요한 기능이다[5].

다. 의존성(dependency)

의존성이란 리소스가 같은 그룹내에서 동작하게 하고, 온라인 또는 오프라인 상태로 만들어주는 두 리소스 사이의 관계를 말한다. 예를 들면 응용 프로그램 리소스는 데이터를 담고 있는 디스크에 의존하고, IP 어드레스 리소스는 네트워크 네임 리소스에 의존한다. 그림 4는 의존성에 대한 예를 나타낸다. 그림에서와 같이 데이터베이스 리소스 DLL은 IP 어드레스 리소스 DLL, 디스크 리소스 DLL에 의존하며, 일반 응용 프로그램 DLL은 디스크 리소스 DLL에 의존한다는 것을 보여준다.

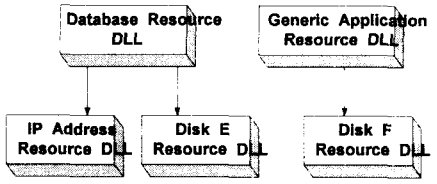

그림 4. 의존성의 예

Fig 4. Dependency example

III. 클러스터 인식 응용 프로그래밍 기법

1. 클러스터 서버의 구조

클러스터에 대한 인터페이스를 제공하는 소프트웨어 컴포넌트는 클러스터 서비스, 클러스터 디스크 드라이버, 클러스터 네트워크 드라이버, 리소스 모니터, 리소스 DLL, 클러스터 관리자, 서버 클러스터API(Applicaion Programming Interface), 클러스터 자동화 서버, 클러스터 데이터베이스로 나누어지며, 각 컴포넌트는 서버 클러스터를 수행하기 위하여 함께 동작한다. 그림 5는 단일 윈도우즈 NT 서버 운영체제에서 서버 클러스터 컴포넌트들간의 관련성과, 여러 응용 프로그램들과의 동작구조를 나타내고 있다.

클러스터를 지원하는 응용 프로그램은 클러스터 API를 통하여 클러스터 서버에 의해 제공되는 특징을 이용할 수 있다. 클러스터를 지원하는 응

용 프로그램은 리소스 모니터에 의해 요청된 상태를 보고하며, 성공적으로 온라인(online) 또는 오프라인(offline)되었는지 요청에 응답한다. 클러스터를 지원하는 응용 프로그램은 응용 프로그램을 책임지고 있는 개발자들에 의해 만들어진 리소스 유형인 커스텀 자원유형으로서 클러스터 서비스에 의해 관리될 수 있다. 커스텀 리소스 유형을 만들기 위해서는 개발자들은 두 가지 DLL을 만들어야만 한다. 즉 리소스 DLL과 클러스터 관리 확장 DLL이다[6].

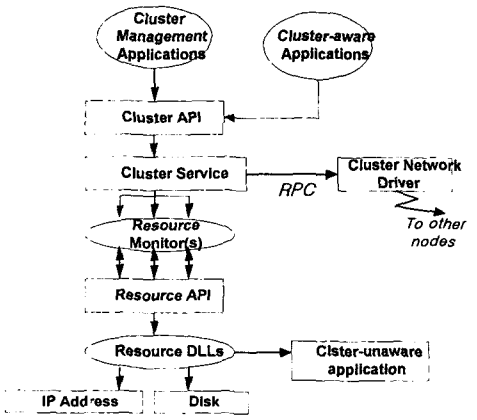

그림 5. 클러스터 서버 구조

Fig 5. Cluster server architecture

가. 클러스터 API

클러스터 API는 클러스터 서비스를 통하여 클러스터가 오브젝트를 관리하도록 한다. 그림 6은 응용 프로그램이 어떻게 클러스터 API를 접근하는지를 보여주고 있으며, 빗금친 부분은 2노드 클러스터의 한 노드를 나타내고 있다. 한 노드에서는 클러스터 서비스, 몇몇 리소스 DLL, 하나 이상의 클러스터 인식 응용 프로그램이 동작하고 있으며, 나머지 노드에서는 클러스터 관리자와 같은 클러스터 관리 응용 프로그램, 클러스터 인식 응용 프로그램이 있다.

클러스터 인식 응용 프로그램은 사용자 리소스를 참조함으로써 지원될 수 있기 때문에 리소스 유형을 반드시 정의해야 한다. 그리고 앞에서 명시한 바와 같이 사용자 응용 프로그램이 사용자 리소스로서 관리되기 위해서는 두가지 DLL을 제공해야 한다. 리소스 DLL과 클러스터 관리자 확장 DLL이 그것이다.

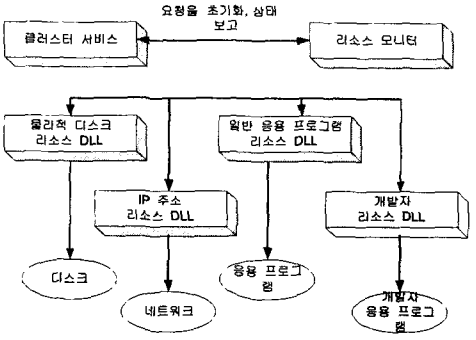

그림 6. 리소스 콘트롤 흐름

Fig 6. Resource control flows

나. 클러스터 관리 확장 API

클러스터 관리 확장 API는 사용자 리소스 유형이 클러스터 관리자에 의해 관리될 수 있도록 COM(Component Object Model) 인터페이스와 함수를 정의한다. 인터페이스는 클러스터 관리 확장 DLL에 의해 사용되며, 함수는 클러스터 관리자와 통신하는 데 이용된다. 윈도우즈 클러스터에 의해 제공되는 사용자 리소스 유형과 일반 리소스 유형은 클러스터 확장 DLL을 통해 일관된 사용자 인터페이스로 관리될 수 있다.

클러스터 관리 확장 API는 클러스터 관리 확장 인터페이스와 클러스터 등록 함수의 정의를 포함하는 cluadmex.idl, cluadmex.h, cluadmex.tlb, cluadmex.lib로 구성된다.

2. 구현환경

일반적인 응용 프로그램은 클러스터 환경의 확장성 및 여러 가지의 장점을 이용하는 데 한계가 있고, 클러스터 서비스가 제공되지 않는 응용 프로그램은 클러스터 서비스를 이용하지 못한다. 그래서 본 논문에서는 이러한 응용 프로그램을 사용자 정의 클러스터 인식 응용 프로그램이라고 규정한다. 즉, 클러스터가 모든 서버 기반의 소프트웨어에게 가용성과 관리의 편의성을 제공하지만, 클러스터 기능의 응용 프로그램은 클러스터 환경의 확장성을 완전히 활용할 수 있다. 클러스터 기능이 없는 서버 응용 프로그램이 클러스터 기능을 갖추게 되면 응용 프로그램의 자동적인 작업인계 기능, 신속한 복구, 쉬운 관리의 편의성과 같은 장점을 가지게 된다. 그래서 본 논문에서는 클러스터 기능이 없는 TCP/IP 소켓 응용 프로그램이 클러스터 환경에서 동작할 수 있도록 사용자 정의 클러스터 인식 응용 프로그램을 구현하기 위해 그림 7과 같이 시스템을 구성하였다.

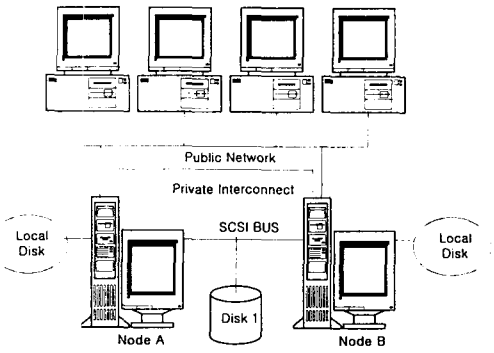

그림 7. 클러스터의 구성
Fig 7. Cluster configuration

(1) 두 개의 윈도우 NT 엔터프라이즈 에디션 운영체제 : OSDB_SERV, OSDB_BACK
(2) 두 노드를 연결하는 하나 이상의 스카시 버스 : Adaptec AHA2940U2W(AIC-7890A)
(3) 스카시 버스에 연결되는 두 개의 하드 디스크 : Seagate ST-34555W
(4) 각 노드에 할당된 CPU: Pentuin Pro 200
(5) 노드 사이의 통신에 필요한 Private 상호연결 : 이더넷 케이블
(6) 클라이언트가 클러스터에 접속하는 데 필요한 Public Network : 이더넷 케이블
(7) Private Interconnect와 Public Network에 필요한 프로토콜 : TCP/IP

3. 구현 방법

본 논문에서는 클러스터 인식 응용 프로그램의 GUI 및 프로그래밍 개발 도구로써 비주얼 C++ 5.0을 사용하였고, 비주얼 C++로 간단한 TCP/IP 소켓 응용 프로그램을 실제 구현하고, 구현된 실제 소켓 응용 프로그램과 비주얼 C++ 5.0과의 연계를 위해서 마이크로소프트사의 클러스터 SDK(Software Development Kit)를 이용하여, 다음과 같은 작업을 수행하였다.

(1) 클러스터를 지원할 응용 프로그램을 만든다.
(2) 응용 프로그램에 클러스터 기능을 첨가한다.
(3) 클러스터 서버가 클러스터 리소스로서 응용 프로그램과 서비스를 관리하도록 새로운 리소스 DLL을 만든다.
(4) 클러스터 서버가 커스텀 리소스 유형을 조정하도록 클러스터 관리 확장 DLL을 만든다.
(5) 리소스가 클러스터를 인식할 수 있도록 리소스 DLL을 클러스터 서버에 등록한다.
(6) 클러스터의 동작여부를 확인하기 위하여 작업 인계를 시험한다.

클러스터 인식 응용 프로그램은 클러스터 노드에서 동작하고 클러스터 리소스로서 관리될 수 있는 응용 프로그램이다. 클러스터 인식 응용 프로그램은 클러스터가 제공하는 환경을 인식하고 클러스터의 특징을 이용하도록 설계된다. 본 논문에서는 구현된 클러스터 인식 응용 프로그램을 직접 실험하기 위해서 서버, 클라이언트가 서로 채팅을 할 수 있는 응용 프로그램을 만들었다.

4. 클러스터 기능의 첨가

클러스터 서비스에서 리소스는 관리될 수 있는 물리적 또는 논리적 컴포넌트이며, 그 종류로는 디스크, 네트워크 이름, IP 주소, 데이터베이스, 웹 사이트, 응용 프로그램, 그리고 온라인 또는 오프라인 될 수 있는 개체가 있다. 리소스는 유형(type)으로서 구성되며, 디스크 드라이버와 같은 물리적 하드웨어와 IP 주소, 파일 공유, 일반 응용 프로그램과 같은 논리적 항목을 포함한다.

클러스터 인식 응용 프로그램이 클러스터 노드에서 동작하고 있다는 것이 감지되면, 클러스터 인식 응용 프로그램은 클러스터 API를 통하여 유용한 작업을 제공한다. 응용 프로그램에 적합한 리소스 DLL을 만듦으로써, 클러스터 서버가 제공하는 더 나은 제어와 유동성을 제공할 수 있다. 커스텀 리소스 유형을 만듦으로써 응용 프로그램에 적합한 방식으로, 리소스 모니터가 자신의 응용 프로그램을 관리할 수 있도록 한다. 또한, 리소스 API를 이용하여 리소스 DLL을 만들 수 있고, 클러스터 관리자가 응용 프로그램을 관리할 수 있도록 클러스터 관리 확장 DLL을 만들 수 있다. 리소스 DLL은 응용 프로그램과 리소스 모니터 사이의 통신을 제어하기 위해 리소스 API에 정의된 엔트리포인트 함수의 실행을 담고 있다.

클러스터 확장 DLL은 응용 프로그램이 클러스터 관리자에 의해 관리될 수 있도록 클러스터 확장 API의 인터페이스 메소드를 이용한다. 클러스터 확장 DLL은 네트워크 관리자의 작업을 단순화시킨다. 즉 응용 프로그램과 독립된 리소스, 그룹에 통합된 리소스를 사용자 인터페이스를 통하여 서로 동작시키도록 한다.

5. 리소스 DLL과 관리 DLL의 설계 및 구현

클러스터 인식 응용 프로그램과 관련된 새로운 리소스 유형을 만드는 것은 가능하다. 새로운 리소스 유형을 만들 때 네트워크 관리자는 클라이언트가 접근하는 가상서버를 구성할 네트워크 이름과, 그것과 관계된 리소스를 새로운 유형과 결합시켜야 한다. 네트워크 관리자가 그룹을 형성할 때, 필요한 리소스들에 대한 의존성이 결여되었다면 클러스터 관리 확장 DLL의 상태를 점검하고, 리소스 DLL을 적재함으로써 같은 그룹 내의 필요한 모든 의존성을 위치시켜야 한다. 예를 들면, 새로운 리소스 유형이 데이터 파일에 관련된 클러스터 응용 프로그램을 관리한다고 가정하면, 네

트워크 관리자는 이 응용 프로그램과, 응용 프로그램의 데이터 파일과 관련된 네트워크 네임 리소스, IP 어드레스 리소스, 공유 디스크 리소스를 형성해야 한다[7].

가. 리소스 유형 마법사

새로운 리소스 유형을 만들기 위해서 그림 8과 같이 비주얼 C++의 "Cluster Resource Type AppWizard"를 이용하여 리소스 이름을 Test라고 명명하고 그림 9와 같은 "Resource Type AppWizard"를 이용한다.

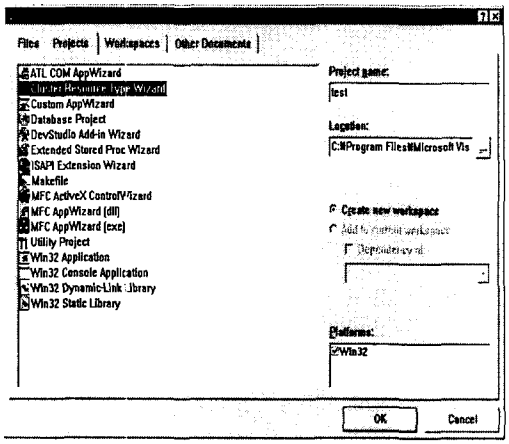

그림 8 . 클러스터 리소스 유형 마법사
Fig 8. Cluster resource type appwizard

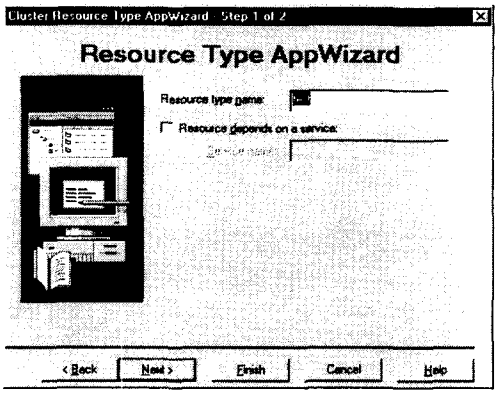

그림 9. 리소스 유형 응용 프로그램 마법사
Fig 9. Resource type appwizard

나. 파라미터 첨가

파라미터는 리소스 유형에 관계된 프라이비트 (private) 속성이다. 네트워크 관리자가 새로운 리소스 유형을 만들기 위하여 클러스터 관리자를 동작시킬 때, 클러스터 관리 확장 DLL은 이 속성(property)에 대한 값을 리턴한다. 이 값들은 클러스터 데이터베이스에 저장된다. 이것은 엔트리포인트 함수와 인터페이스 메소드의 뼈대를 만든다. 그림 10에서 파라미터 이름을 MaxUser라고 주었고, Maxium value를 10이라고 주었는데 이것은 서버쪽에서 최대 사용자를 얼마나 받아들일 것인지를 나타낸다.

이렇게 해서 클러스터를 지원하는 리소스에 대한 DLL, 즉 "cluster resource DLL"과 "cluster administration DLL"이 만들어진다[8].

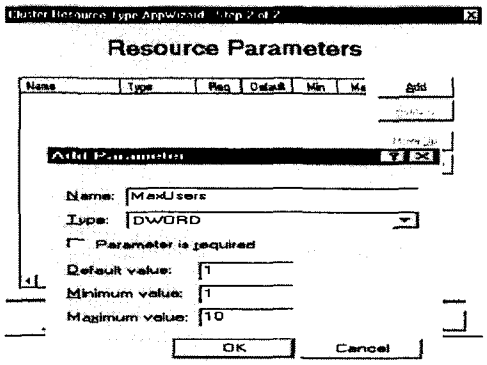

그림 10. 리소스 파라미터 첨가
Fig 10. Add resource parameter

다. 헤더 파일 첨가

응용 프로그램이 클러스터를 인식하게 하기 위해서는 그림 11과 같이 응용 프로그램에 대한 정보를 담고 있는 헤더 파일, 즉 <server.h>, <sversok.h>를, 클러스터 행위를 정의하고 있는 <clusapi.h>, <resapi.h>, <stdio.h>다음에 첨가시켜야만 한다. 그리고 그림 12에서 보는 바와 같이 리소스에 관계된 IP 어드레스, 네트워크 이름, 공유 이름을 의존적인 관계를 성립하기 위해 파라미터에 추가시켰다.

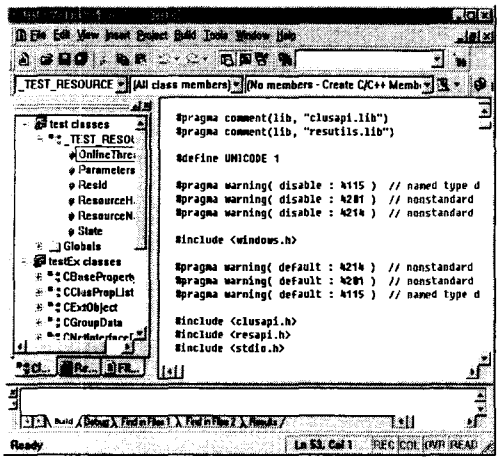

그림 11. 리소스 DLL과 관리 DLL

Fig 11. resource DLL and administration DLL

```
//
// Type and constant definitions.
//

#define TEST_RESNAME  L"TEST Sample"
#define TEST_SVCNAME  TEXT("SocketServer")

#define DBG_PRINT printf

// ADDPARAM: Add new parameters here.
#define PARAM_NAME__SHARENAME L"ShareName"
#define PARAM_NAME__PATH L"IPAddress"
#define PARAM_NAME__REMARK L"NetName"


// ADDPARAM: Add new parameters here.
typedef struct _TEST_PARAMS {
    PWSTR           ShareName;
    PWSTR           IPAddress;
    PWSTR           NetName;
} TEST_PARAMS, *PTEST_PARAMS;
```

그림 12. 파라미터 정보

Fig 12. Paramater information

IV. 실험 결과 및 고찰

본 논문에서 구현한 클러스터 인식 응용 프로그램에 대한 리소스는 그림 13의 "Resource DLL"과 클러스터 레지스터에 등록된 "Cluster Administration Extension DLL"이다. 구현한 DLL은 "regcladm.exe"라는 클러스터 서버 응용 프로그램으로 클러스터 서버에 리소스 유형으로 등록하였고, 그룹을 형성할 의존성에 대한 리소스로서 네트워크 이름, IP address, 디스크 리소스를 만들었다. 실제 구현한 DLL의 동작을 확인하기 위하여 서버 채팅 프로그램을 수행하고 있는 OSDB_SERVER의 SOCKET_GROUP을 그림 14와 같이 수동적으로 클러스터 내의 나머지 서버로 이동시켰을 때, 수행되던 서비스는 그림 15와 같이 자동적으로 OSDB_BACK으로 작업인계가 이루어진다는 것을 확인할 수 있었다. 그리고 작업인계 시간은 약 5초 정도에서 이루어진다는 것을 알 수 있었다.

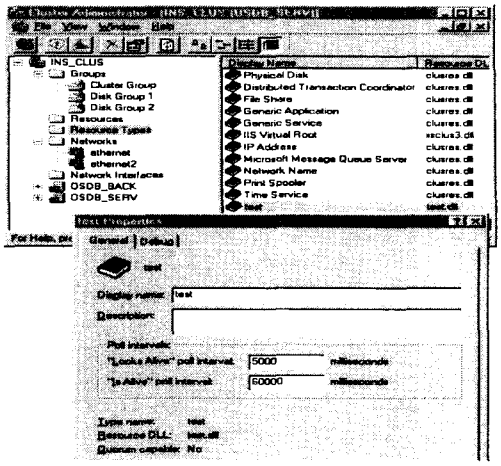

그림 13. 클러스터 인식 응용 프로그램

Fig 13. Cluster-aware application

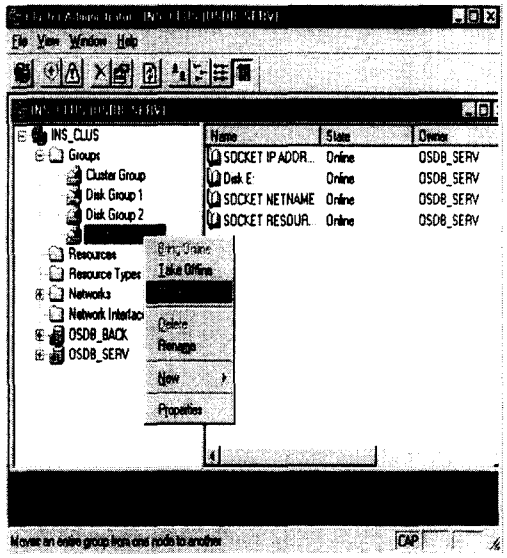

그림 14. 결함 전의 응용 프로그램

Fig 14. Application before failure

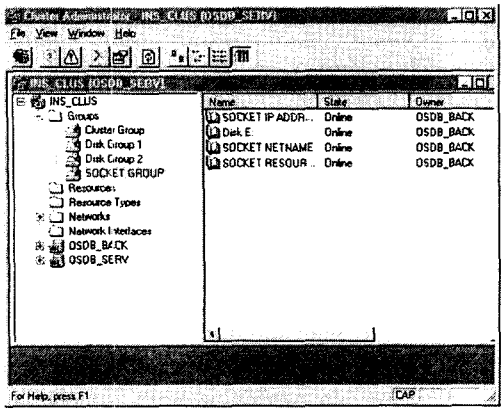

그림 15. 결함 후의 응용 프로그램
Fig 15. Application after failure

V. 결 론

현재 데이터베이스 서버와 웹 서버, 메일 서버와 같이 서버의 중요성이 높은 경우에는 클러스터의 작업인계 기능을 지원하고는 있지만 모든 서버 응용 프로그램이 클러스터 기능을 지원하지는 않는다.

본 논문에서는 응용 프로그램의 자동적인 작업인계 기능, 신속한 복구, 관리의 편의성과 높은 확장성을 위하여 클러스터 인식 응용 프로그램의 구현기법에 관하여 기술하였다. 이를 위하여 간단한 TCP/IP 소켓 응용 프로그램을 구현하고, 클러스터를 지원하게 함으로써 실제 마이크로소프트 클러스터 서버 환경에서 한 서버 내의 응용 프로그램이 다운되는 경우 다른 서버가 자동적으로 다운된 서버 내의 응용 프로그램을 작업 인계한다는 것을 확인할 수 있었다.

클러스터가 대부분의 서버 응용 프로그램의 가용성과 관리의 편의성을 제공하지만, 클러스터 인식 응용 프로그램은 클러스터 환경의 확장성을 완전히 활용할 수 있다.

향후 연구로는 클러스터가 지원하는 노드의 수를 늘리고 확장성이 개선되어야 할 것이며, 웹 서버와 데이터베이스 서버가 이상이 발생했을 경우 빠른 서비스를 제공하기 위해서는 작업인계 시간이 향상되어야 할 것이다. 또한 보다 편한 GUI (Graphic User Interface) 환경을 지원하기 위한 OCX(Ole Control eXtensions)와 같은 컴포넌트를 이용한 클러스터 오토메이션 서버가 개발되어야 할 것이다.

참고문헌

[1] HTTP://www.digital.co.kr 디지털 매거진, Windows NT용 디지탈 클러스터 백서, *1996. 11.*

[2] Elizabeth Clark, "Server Clustering: The Good of the Many", *Network, August. 1997.*

[3] Rob Short, Rod Gauche, John Vert and Mike Massa, "Windows NT Clusters for Availability and Scalabilty", *Microsoft Corporation, 1998.*

[4] Lori Merric, "Clustering Support for Microsoft SQL Server", *Microsoft Corporation, 1997.*

[5] Vogel W, Dimitri D, Agrawal A, Chia T, and Guo K, "Scalability of the Microsoft Cluster Service", *Proceedings of the Second Usenix Windows NT Symposium, Seattle, WA, pp 1-4, August 1998.*

[6] Werner Vogels, "the Design and architecture of the Microsoft Cluster Service", *Microsoft Corporation, 1997.*

[7] Sharon J, "Writing Microsoft Cluster Server (MSCS) Resource DLLs", *Microsft Corporation, 1997.*

[8] Scott H. Davis, "Clusters for Windows NT", *Microsoft Developer's Network Library, 1998.*