분산 공유메모리를 기반으로 한 서브 클러스터 그룹의 자료전송방식

이 기 준†

요 약

최근 네트워크 기술의 비약적인 발전은 고속 그리고 저가의 클러스터 시스템을 구축할 수 있는 기본 토대를 제공하여 주었다. 이러한 기존 클러스터 시스템은 안정화된 고속의 지역 네트워크를 기반으로 일정 수준의 시스템으로 구성되는 것이 일반적인 경향이다. 본 논문에서 제안하는 다중 분산 웹 클러스터 그룹은 개방 네트워크상에 존재하는 저가, 저속의 시스템 노드를 대상으로 하여, 주어진 작업에 대한 병렬수행 및 SC-Sever의 공유메모리를 통한 효율적인 작업 분배와 시스템 노드간의 상호 협조 작업을 통하여 고성능, 고효율 그리고 고가용성을 얻을 수 있는 웹 클러스터 모델이다. 이를 위하여 다중 분산 웹 클러스터 그룹은 복수개의 시스템 노드를 단일한 가상 네트워크로 묶어 놓은 서브 클러스터 그룹으로 구성하고, 서브 클러스터 그룹내의 효율적인 자료전송을 위하여 분산 공유 메모리를 이용한다. 제안된 모델은 사용자로부터 요구되는 대규모의 작업에 대하여 분산 공유 메모리를 기반으로 한 부하분배 및 병렬 컴퓨팅 방식을 이용하므로 처리 효율을 높일 수 있다.

A Data Transfer Method of the Sub-Cluster Group based on the Distributed and Shared Memory

Kee-Jun Lee†

ABSTRACT

The radical development of recent network technology provides the basic foundation which can establish a high speed and cheap cluster system. It is a general trend that conventional cluster systems are built as the system over a fixed level based on stabilized and high speed local networks. A multi-distributed web cluster group is a web cluster model which can obtain high performance, high efficiency and high availability through mutual cooperative works between effective job division and system nodes through parallel performance of a given work and shared memory of SC-Server with low price and low speed system nodes on networks. For this, multi-distributed web cluster group builds a sub-cluster group bound with single imaginary networks of multiple system nodes and uses the web distributed shared memory of system nodes for the effective data transmission within sub-cluster groups. Since the presented model uses a load balancing and parallel computing method of large-scale work required from users, it can maximize the processing efficiency.

키워드 : 클러스터 시스템(Cluster System), 분산 컴퓨팅(Distributed Computing), 분산 공유 메모리(Distributed Shared Memory)

1. 서 론

최근 네트워크 기술의 급속한 성장과 마이크로 프로세서의 비약적인 발전은 초고속 클러스터 시스템을 구축할 수 있는 기본 토대를 마련하여 주었다.

고속의 네트워크에 연결된 클러스터 시스템은 슈퍼컴퓨터나 메인 프레임에 비하여 매우 저렴한 비용으로 주어진 작업을 병렬처리 할 수 있다는 장점을 지니고 있으며, 성능이 뛰어나고 가용성이 높은 시스템을 구축하는데 용이하다. 또한 단일 프로세서 시스템(single processor system)이나 엄격한 다중 프로세서 시스템(multi processor system)에 비하여 확장성이 뛰어나 적은 투자비용에 비하여 효율성과 신뢰성을 가지는 시스템을 구축할 수 있다[1].

이러한 클러스터 시스템들은 PVM(parallel virtual machine)[2], MPI(message passing interface)[3]와 같은 표준화된 메시지 전달 기법을 사용하여 여러 시스템을 단일한 클러스터 환경으로 구축할 수 있는 병렬 프로그래밍 모델 환경을 제공하여 주며, 기후 모델링, 시각 및 인지, 반도체 모델링 등 다양한 분야에서 단일 프로세서 시스템이 지닌 물리적인 한계를 해결해 줄 수 있는 대안으로 사용되고 있다[4].

클러스터 모델은 구성 및 적용분야에 따라 HPC(high performance cluster)[5], Bulk Storage 클러스터[6], Web/Internet 클러스터[7], HA(high availability) 클러스터[8]로 구분 지을 수 있다. HPC 클러스터는 단일한 고성능 CPU에

† 정 회 원 : 광주보건대학 컴퓨터보안과 교수
논문접수 : 2003년 8월 4일, 심사완료 : 2003년 11월 27일

서 수행하기 불가능한 대규모 연산을 처리하기 위한 고속의 클러스터 시스템으로 다수의 시스템을 고속의 네트워크로 연결한 구조를 지니고 있다[9]. HPC의 대표적인 BEOWULF 시스템[10]은 16 node 클러스터를 리눅스와 표준 소프트웨어 패키지를 이용하여 개발한 시스템으로 일반 PC와 Linux를 사용하여 모델의 확장과 시스템 변경이 용이한 특징을 지니고 있다. 또한 Bulk Storage 클러스터 시스템은 고속의 네트워크를 이용하여 다수 시스템간의 데이터 저장과 서비스를 공유하기 위한 클러스터 시스템이며, Web/Internet 클러스터 시스템은 다중의 사용자로부터 요청된 서비스 작업을 원활히 수행하기 위한 LB(load balancing) 클러스터 시스템으로, 가상서버를 중심으로 실제 작업을 수행하는 서버들이 연결되어 있어 확장성과 유용성이 높은 시스템이다[11, 12].

HA 클러스터는 구성된 시스템 상에서 발생할 수 있는 시스템 장애 문제를 해결하기 위한 고가용 클러스터 기술로, 시스템의 장애나 재구성 요구를 적절히 감지하며, 클러스터에 남아있는 노드로 부하분배를 수행하여 무정지, 무제한의 서비스를 제공하여 준다[13].

본 논문에서는 기술한 기존 클러스터 시스템 기술을 바탕으로 인터넷 기반의 다중 분산 웹 클러스터 모델을 새로이 구축하고, 구축된 모델에서 사용자가 요청한 서비스 작업을 효율적으로 분산 처리할 수 있는 분산 공유 메모리기반의 자료 전송방식에 관하여 기술하고자 한다.

기존 클러스터 시스템의 메시지 전달방식과 공유 메모리 구조는 고속 네트워크 환경에서 다중 프로세서나 다중 프로그램과 같은 공유 주소 공간 프로그램이 가능하며 각 노드간의 메시지 전달방식이 자유로이 이용될 수 있다. 그러나 제안 모델은 개방된 네트워크 상에 존재하는 다양한 시스템 노드를 대상으로 하므로 이를 적용하는 경우 각 노드간의 상이한 통신과 동기화방식으로 인하여 전 기능들을 활용하기 힘들고, 노드 내부간의 병렬성을 동시에 충족시키기 어렵다. 따라서 제안한 다중 분산 웹 클러스터 모델에서는 시스템 노드 상호간의 효율적인 자료와 신호 전송을 위하여 분산 공유 메모리기반의 서브 클러스터 그룹을 구축한다. 복수개의 서브 클러스터 그룹으로 구성된 다중 분산 웹 클러스터 모델은 사용자가 요구하는 대규모의 작업을 분산 공유메모리 방식을 기반으로 한 부하분배 및 병렬 연산을 이용하므로 처리의 효율을 높일 수 있다.

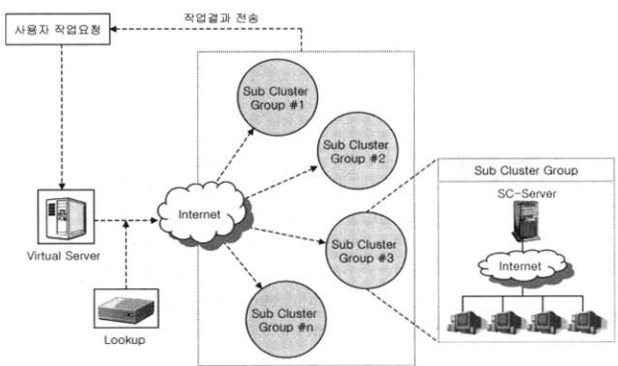
본 논문은 다음과 같이 구성되어 있다. 먼저 2장에서는 다중 분산 웹 클러스터 그룹의 형태와 서브 클러스터 그룹의 구성방식에 대하여 기술하고, 3장에서 분산 공유 메모리를 이용한 서브 클러스터 그룹내부의 자료전송방식에 대하여 기술한다. 그리고 4장의 구현 및 모의실험을 통하여 제안방식의 유용성에 대하여 고찰하고, 마지막 5장에서 결론을 맺는다.

2 다중 분산 웹 클러스터 모델

다중 분산 웹 클러스터 모델은 시스템 노드의 개별적 성능의 차이를 극복하기 위하여 먼저 복수 개 시스템 노드를 단일한 가상 네트워크에 묶어놓은 서브 클러스터(Sub Cluster) 그룹으로 구성하고, 이중 임의의 한 시스템노드를 선택하여 서브 클러스터 그룹을 대표하는 SC-Server로 지정한다. 이러한 방법으로 구성된 서브 클러스터 그룹은 다중 분산 웹 클러스터 모델을 구성하는 단위모듈이 된다.

2.1 다중 분산 웹 클러스터 모델 구성

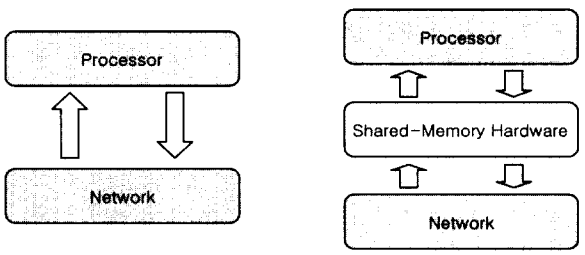
다중 분산 웹 클러스터 모델은 단일한 가상 네트워크에 묶여져 있는 서브 클러스터 그룹과 서브 클러스터 그룹의 집합체인 다중 분산 웹 클러스터 모델로 구분한다. (그림 1)과 같이 서브 클러스터 그룹은 일정 수량의 시스템 노드들을 동적(dynamic)으로 구성하고, 가상 서버(virtual server)로부터 전송되어온 사용자의 서비스 작업을 분산 처리할 수 있는 병렬 컴퓨팅 구조로 구성되어 있으며, 다중 분산 웹 클러스터 모델은 이러한 서브 클러스터 그룹들에 대한 작업의 지시와 수행결과의 통합을 수행한다.



(그림 1) 다중 분산 웹 클러스터 모델의 구성도

가상서버에 의해 묶여진 서브 클러스터 그룹은 네트워크 상에 분산되어 있는 여러 시스템 노드들을 대표하는 SC-Server에 의해 하나의 노드로 그룹화 되어있고, 이를 외부에서 바라볼 때 서브 클러스터 그룹은 한 개의 노드로 구성된 단일 시스템으로 보여 지게 된다. 따라서 가상서버와 SC-Server 간의 연결은 단일 네트워크 상에 묶여진 추상화된 내부 네트워크로, SC-Server와 시스템 노드간은 개방 네트워크 상에 연결된 외부 네트워크로 인식하게 된다. 이러한 구조적 특성상 가상서버와 SC-Server, SC-Server와 시스템 노드간의 자료전송방식은 각기 다른 구조로 이루어지게 된다. 일반적인 자료전송방식은 메시지 전달방식(그림 2)(a)과 공유 메모리 방식(그림 2)(b)으로 구분된다. 서브 클러스터 내부의 각 시스템 노드들은 개방 네트워크 상에 무작위로 분포되어 있기 때문에 이들과 SC-Server간의 자

료와 제어의 전송은 메시지 에이전트를 이용한 전송방식으로 이용하며, 서브 클러스터 그룹의 SC-Sever와 가상서버 간의 자료전송은 분산 공유 메모리 방식을 이용한다.



(a) 메시지 전달방식 (b) 공유 메모리 방식

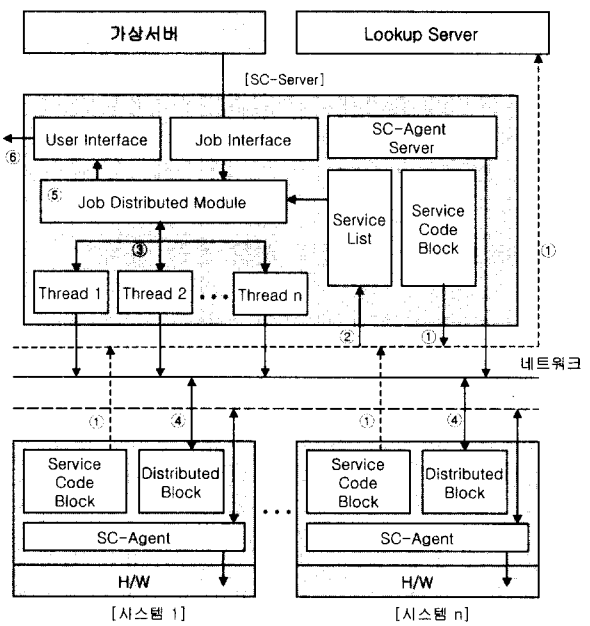
(그림 2) 메시지 전달과 공유메모리 방식

2.2 서브 클러스터 그룹

2.2.1 서브 클러스터 그룹의 구성

가상서버의 서비스 수행요구에 의해 구성된 서브 클러스터 그룹은 내부의 시스템 노드 중 한 개의 노드를 임의 선출해 SC-Server의 임무를 부여한다. (그림 3)은 구성된 서브 클러스터 그룹과 가상서버, 전체 시스템 노드의 서비스 모듈을 통합 관리하는 Lookup Server의 구조를 나타내고 있다. SC-Server와 각각의 시스템 노드에 구성되어있는 주요 모듈의 기능은 다음과 같다.

- Job Interface : 가상서버로부터 전송되어온 작업 요구 패킷을 분석하여 사용자의 주소(IP)와 포트(port), 그리고 요청된 작업 내용을 분석한다. 분석된 작업은 Job Distributed Module에 전송되어 병렬 분산 작업을 수행하고, 사용자 주소와 포트는 수행결과를 사용자에게 직접 전송하기 위하여 임시 저장한다.
- Job Distributed Module : Job Interface로부터 전달된 사용자가 요청한 작업 내용을 분석하고, 분석되어진 내용을 토대로 수행할 수 있는 서비스 코드 블록의 목록을 Service List 모듈에 요청한다. 요구한 서비스 코드 블록의 목록을 Service List 모듈로부터 받은 후 서비스 코드 목록의 수(서브 클러스터 노드의 수)에 비례한 쓰레드(thread)를 발생시켜 요청된 작업을 분산 처리한다.
- Service List : Service List 모듈은 Job Distributed Module로부터 요청받은 서비스 코드 블록을 찾기 위하여 Lookup Server에 검색을 요청한다. Lookup Server는 Service Code Bank에 저장된 서비스 코드 블록중 수행 가능한 노드의 목록을 Service List 모듈에 전송하고, Service List 모듈은 전송받은 서비스 코드 목록을 Job Distributed Module에 전달한다.
- Service Code Block(SCB) : Service Code Block은 해당 시스템 노드에서 처리할 수 있는 서비스의 코드로 SC-Server로부터 발생된 쓰레드에 의해서 수행된다. 다중 분산 클러스터 모델의 시스템 노드들은 Service Code Block을 Lookup-Server에 등록하여 수행 가능한 상태를 유지하게 된다.
- Distributed Block : Distributed Block은 SC-Server에서 발생한 쓰레드에 의해 해당 시스템 노드에서 수행되는 서비스 코드 블록이다. SC-Server에 의해 분할되어진 작업의 내용이 쓰레드를 통하여 시스템 노드의 Distributed Block에 전송되어지고, 전송된 작업의 결과값은 SC-Server의 Job Distributed Module에 전송되어진다. 따라서 각 시스템 노드의 Distributed Block은 SC-Server와의 자료전송을 위한 창구역할을 수행한다.



(그림 3) 서브 클러스터 그룹

2.2.2 서브 클러스터 그룹의 작업과정

구성된 서브 클러스터 그룹에 가상서버의 서비스 요구가 요청되었을 때 이를 수행하는 과정은 다음과 같다.

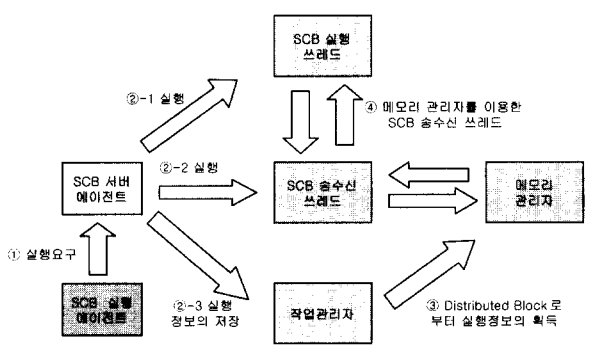
- 서브 클러스터 그룹을 구성하고 있는 각 시스템 노드들은 클러스터 그룹 참여를 위하여 자신의 서비스 코드 블록을 Lookup Server에 등록한다(①). Lookup Server에 등록되어진 코드 블록은 Lookup Server의 Service Code Bank에 저장되어지며, 이후 SC-Server나 다른 시스템 노드의 작업 요청 시 사용하게 된다.
- 가상 서버부터 해당 서브 클러스터 그룹에 서비스 작업이 요청되어지면 Job Interface 모듈은 서비스 요구 패킷을 분석하고, 이중 작업 요청 영역을 Job Distributed Module에 전송한다. Job Distributed Module은 서비스 작업에 필요한 서비스 코드 목록을 전송받기 위하여 Service List 모듈을 통하여 Lookup Server로부터 서비스의 목록 (시스템의 목록)을 전송 받는다(②).
- Lookup Server로부터 전송되어진 서비스 코드 목록은

Job Distributed Module로 전달되고, 수행해야할 작업의 내용이 서비스의 목록에 비례하여 분할되어진다. 분할되어진 각 작업의 내용은 각각 쓰레드에 의해 수행되고, 발생된 쓰레드는 서비스를 수행할 수 있는 시스템 노드의 Distributed Block과 원격 메소드 호출(remote method invocation)을 이용하여 서비스를 제공하는 시스템 노드에서 수행된다(③).

- 서브 클러스터 그룹을 구성하는 각 시스템 노드의 Distributed Block은 SC-Server에서 구동된 쓰레드를 통해 처리해야 할 자료를 전송 받아 작업을 수행한다(④).
- 각 시스템 노드에서 수행된 작업 결과는 SC-Server의 Job Distributed Module에 취합되어진다(⑤).
- SC-Server의 Job Distributed Module에 취합되어진 내용은 Job Interface 모듈에 의해 임시 저장된 사용자의 주소와 포트번호를 이용하여 작업을 요청한 사용자에게 직접 전송되어진다(⑥).

3 서브 클러스터 그룹의 자료 전송 방식

다중 분산 웹 클러스터 모델을 구성하는 SC-Server는 요청된 작업을 시스템 노드의 Distributed Block 모듈에 서 병렬 수행되어지며, 수행결과는 SC-Server의 Job Distributed Module에 취합되어 사용자에게 전송하게 된다. 이를 위하여 SC-Server와 시스템 노드들은 Job Distributed Module의 분산 공유 메모리를 기반으로 한 자료전달방식을 이용한다. (그림 4)는 서비스 코드 블록, Job Distributed Module내의 공유메모리 관리자, 시스템 노드의 Distributed Block내의 작업 관리자간의 전체적인 관계를 나타내고 있다.



(그림 4) SCB, 공유메모리 관리자와 작업관리자간의 관계

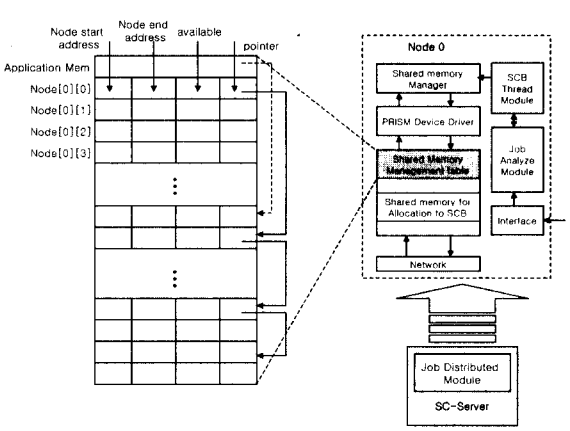
각 모듈별 동작방식은 다음과 같다.

- 서비스 코드 블록(SCB) 서버 에이전트는 시스템 가동 이후 계속 수행중이며, SCB 실행(run) 에이전트는 SC-Server에 의해서 서비스 코드 블록를 수행될 때 실행된다.
- SC-Server로부터 서비스 코드 블록의 수행요청이 있으면, SCB 서버 에이전트는 서비스 코드 블록을 실행한다(①).
- 요구되어진 서비스 코드 블록의 실행을 위하여 SCB 서버 에이전트는 SCB 실행 쓰레드와 SCB 송수신 쓰레드를 수행한다(②-1, ②-2).
- SCB 송수신 쓰레드는 공유메모리 관리자를 이용하여 다른 노드와의 자료 통합과 공유를 수행하며, 작업관리자는 SCB 송수신 쓰레드에 대한 여러 정보들을 저장관리한다(②-3).
- 작업관리자는 공유 메모리 관리자가 동작하기 위해 필요한 정보를 전달한다(③).
- SCB 송수신 쓰레드는 프로그램 초기화단계에서 공유메모리 관리자에 의해 메모리를 할당받고, 일단 공유 메모리를 할당받은 후에 SCB 실행 쓰레드는 공유 메모리를 통해서 통신한다(④).

3.1 공유 메모리 관리자

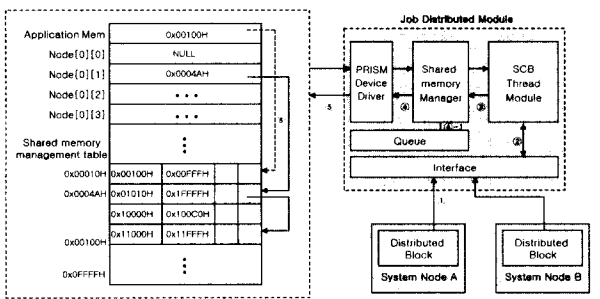
(그림 5)는 SC-Server 내의 공유 메모리 관리자를 포함한 Job Distributed Module의 전체적인 구조이다. Job Distributed Module 구조는 요구된 서비스 작업을 분석하여 다중처리하기 위한 응용프로그램 부분과, 공유 메모리 관리자, PRISM 디바이스 드라이버와 공유메모리들로 구성된 계층적 구조이다.

공유 메모리의 Node 0에서는 다른 시스템 노드와 참조할 수 있도록 공유 메모리 관리(Shared Memory Management) 테이블이 존재하며, 이 공유 매모리 관리 테이블에서 전체 시스템 노드들의 공유메모리 상태를 관리한다. 시스템 노드들에게 공유 메모리를 할당하고자 할 때 메모리의 일관성을 유지하기 위하여 할당된 공유 메모리 공간과 사용가능한 공간을 분리하여 저장한다. 따라서 시스템 노드들이 요구한 메모리는 전체 공유 메모리에서 사용하지 않는 메모리 공간을 할당하여 줌으로써 메모리 일관성을 유지할 수 있다. 만일 시스템 노드에서 메모리 관리자에게 메모리를 요구할 경우, 메모리 관리자는 공유 메모리 관리 테이블을 참조하여 사용 가능한 공간을 할당하고, 만일 이미 할당된 메모리 공간이 사용되지 않을 경우 시스템 노드의 요구에 따라 제거한다.



(그림 5) SC-Server의 Job Distributed Module 구조

(그림 6)은 SC-Server의 Job Distributed Module에 의해서 분산 수행된 시스템노드들이 공유 메모리를 요청하였을 때 Job Distributed Module의 공유 메모리 관리자의 동작방식을 나타낸다. 시스템 노드 A가 공유메모리의 할당을 SC-Server에게 요구하였을 때 SC-Server의 SCB 쓰레드 모듈은 시스템 노드 A의 할당요구를 받아(①) 공유 메모리 관리자에게 현재 할당 가능한 메모리 영역의 검색을 요구한다(②). 공유 메모리 관리자는 PRISM 디바이스 드라이버를 통하여(③) SMM 테이블에서 공유메모리의 상태를 점검하고(④), 만일 할당공간이 남아있을 경우 이를 시스템 노드 A에게 할당한다(⑤, ⑥). 그러나 사용 가능한 메모리 공간이 없을 때에는 시스템 노드 A의 메모리 요청을 대기큐에 입력한 다음 이후에 가능한 공간이 남았을 때, 이를 시스템 노드 A에게 할당한다(④-1). 공유메모리 관리자는 전달된 사용 가능한 공유메모리의 번지를 PRISM 디바이스 드라이버를 통하여 전달받고 이를 시스템 노드 A에게 전송한다.

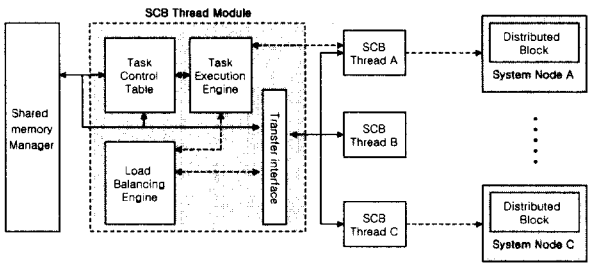


(그림 6) 메모리 요구시 공유메모리의 할당

3.2 SCB 쓰레드 모듈

SCB 쓰레드 모듈은 Job Distributed Module 내에 요구된 서비스 작업을 해당 시스템 노드에서 수행하기 위하여 작업의 분석 및 분석된 작업에 대한 서비스 코드블록 쓰레드를 실행시킨다. 이때 SCB 쓰레드 모듈은 각 시스템 노드에 대하여 발생된 쓰레드를 제어할 수 있으며 또한 해당 시스템 노드에서 수행되는 연산작업에 대한 공유 메모리 관리를 동시에 수행한다. SCB 쓰레드 모듈에 의해 실행되는 쓰레드는 각각 작업명이 주어져 이 작업명에 의해 상대방의 간섭없이 독립적인 작업을 수행한다. 이러한 다중 처리 방식은 프로세서의 처리량을 증가시키는 이점뿐 아니라 작업의 대기시간을 줄일 수 있다. 또한 SCB 쓰레드 모듈은 임의의 시스템 노드에 대한 작업수행이 많아질 경우 전체 서브 클러스터 모델의 성능이 악화되어지는 문제를 해결하기 위하여 시스템 노드의 작업수행여부를 점검하여 각 시스템 노드에서 균등하게 작업이 수행될 수 있도록 작업 분배방식을 이용한다. SCB 쓰레드 모듈은 시스템 노드들이 공유 메모리를 통해 통신할 수 있도록 시스템 노드에 대한 작업의 정보를 공유 메모리 관리자에게 제공하며, SCB 서버 에이전트가 SCB 실행 에이전트의 요구를 받았을 때 SCB 실행 쓰레드와 SCB 송수신 쓰레드를 동시에 수행시킨다. SCB 송수신 쓰레드가 공유 메모리에 의한 자료전송을 위해서 공유 메모리 관리자가 수행되면, SCB 쓰레드 모듈에게서 시스템 노드 번호, 작업명 등의 정보를 얻는다.

(그림 7)에서 SCB 쓰레드 모듈은 작업 제어 테이블(Task Control Table), 부하 균등엔진(Load Balancing Eengine), 작업 수행 엔진(Task Execution Engine)와 전송 인터페이스(Transfer Iinterface)로 구성된다.



(그림 7) SCB 쓰레드 모듈의 구조

- 작업 제어 테이블(TCT) : 작업 제어 테이블은 현재 수행중인 시스템 노드의 작업수행에 대한 정보를 관리하는 테이블이다. TCT는 다른 시스템 노드로부터 해당 노드 쓰레드에 대한 정보의 요구가 있을 때 이를 제공하여준다. 제공하는 정보는 해당 시스템 노드에서 수행중인 작업의 상태, 작업명, 작업을 수행중인 쓰레드의 명, 전체작업에 대한 시스템 노드의 수와 서브 클러스터 그룹내에서 수행중인 쓰레드의 이름 등이다.
- 전송 인터페이스(TI) : 전송 인터페이스는 현재 수행중인 쓰레드를 통하여 해당 시스템의 부하여부, 시스템 상태, 수행 작업의 수등의 정보를 수집하여 부하 균등 엔진(LBE)에게 전달하여 수행해야할 서버스 작업에 대한 부하 균등 정보를 제공하여준다. 따라서 TI는 모든 시스템 노드에게 시스템 부하정보를 요구하는 메시지를 방송하고 이를 통해 받은 정보를 LBE에게 전달한다. 또한 부하 균등 처리된 정보를 선택된 시스템 노드의 쓰레드에게 전달한다.
- 부하 균등 엔진(LBE) : 부하 균등 엔진은 TI로부터 받아들인 해당 시스템 노드의 상태를 분석하여 부하의 부담이 적은 노드를 선택하여 이를 통하여 서브 클러스터 그룹의 전체적인 성능향상을 꾀한다. TI로부터 받아들인 해당 시스템들의 성능, 자료 전송률, 현재 데이터 처리 여부등 정보를 분석하여 시스템 노드의 특징값(SC_{cn})으로 결정되고, 결정된 시스템 노드의 특징값을 기준으로 작업분배를 수행한다.
- 작업 수행 엔진(TEE) : 작업 수행 엔진은 전송 인터페이스를 통하여 부하 균등 처리된 시스템 노드들에게 실행

명령을 전달하는 역할을 수행한다. 이때 TEE는 TCP/IP 프로토콜을 기반으로 실행명령을 해당 시스템 노드에 전송한다.

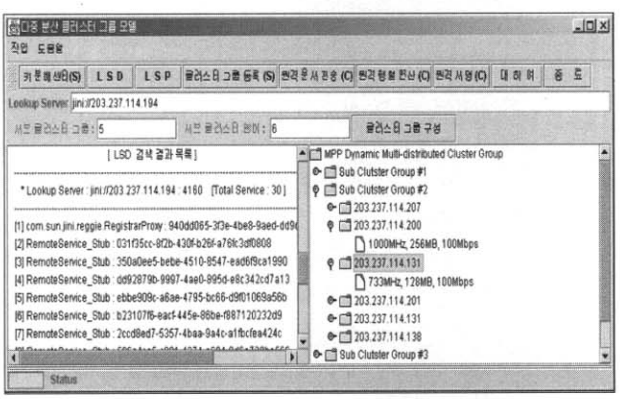
4 구현 및 실험

본 장에서는 제안한 모델의 구현 및 실험을 통하여 시스템의 효율성 및 확장 가능성, 유용성에 대하여 실험하고자 한다.

4.1 서브 클러스터 그룹의 동적 구성

다중 분산 클러스터 그룹을 구현하기 위하여 리눅스 가상 서버는 커널 2.0.35를 사용하였고, IP Tunneling 방식을 이용하여 사용자의 서비스 요구를 해당 서브 클러스터 그룹이 직접 전송할 수 있도록 구성하였다. 서브 클러스터 그룹의 서비스 코드 블록의 등록 및 요청, 분산 서비스 작업 등의 모듈은 SUN의 JDK 1.3.1을 이용하여 작성하였다. 실험에 참여한 시스템 노드들은 총 32대로 Linux와 Window 운영체제 환경의 시스템이며 각 시스템 노드들은 모두 고유 IP를 가지고 인터넷에 연결되어 있다.

(그림 8)은 전체 시스템 노드에 대하여 동적 구성된 서브 클러스터 그룹의 목록을 SC-Sever에서 수행한 것이다.



(그림 8) 서브 클러스터 그룹의 동적 구성

(그림 8)은 총 30개의 서비스 코드 블록을 등록한 시스템 노드들에 대하여 5개의 서브 클러스터 그룹을 구성하고 구성된 각 서브 클러스터 그룹에 대하여 6대의 시스템 노드를 배치하였을 때의 상황을 보여주고 있다. 각 서브 클러스터에 배치되는 시스템 노드들은 임의선정을 통하여 무작위 배치된다. 따라서 여러 서브 클러스터 그룹에 중복 배치되는 시스템이 존재할 가능성이 있다. 구성된 서브 클러스터 그룹의 우선순위는 클러스터 특징값(SC_{cn})과 현재 수행중인 작업의 수에 의해 결정지어 진다. <표 1>은 구성된 서브 클러스터 그룹에 대한 시스템 배치현황과 특징값을 나타낸다.

<표 1> 각 서브 클러스터 그룹의 구성시스템 및 특징값(SC_{cn})과 우선순위

클러스터 그룹	Sub Cluster1	Sub Cluster2	Sub Cluster3	Sub Cluster4	Sub Cluster5
참여 시스템 (203.237.114)	139 146 132 200 143 133	207 200 131 201 131 138	197 143 198 146 196 138	137 143 135 197 134 133	139 133 134 207 206 139
특징값 (SC_{cn})	2721.297	2754.8	2699.4	2599.4	2549.4
우선순위	②	①	③	④	⑤

4.2 서브 클러스터 그룹의 효율성

본 모의실험에서 구축한 다중 분산 클러스터 그룹은 <표 2>와 같이 최대 6대의 시스템 노드로 구성된다.

<표 2> 서브 클러스터 그룹의 시스템 노드 사양

No	IP	CPU	RAM(MB)	Network(Mbps)	OS
1	203.237.114.48	P3-1000B	256	10/100	WinXP
2	203.237.114.194	P-433	96	10/100	Win98
3	203.237.114.195	P-600	64	10/100	Win98
4	203.237.114.199	P2-1.8G	128	10/100	winXP
5	203.237.114.200	P3-1.4G	256	10/100	Win2000
6	203.237.114.203	P3-1G	128	10/100	WinXP

4.2.1 시스템 노드에 따른 작업량 분배

100×100, 200×200, 400×400, 800×800 행렬을 대상으로 연산을 요청하였을 때의 작업분배 과정을 실험하였다. 실험을 통하여 얻어진 평균 수행시간을 해당 실험의 대표값으로 사용하였다. 실험의 내용은 서브 클러스터 그룹의 시스템 노드에 일괄적인 작업분배를 했을 때의 작업소요시간과 분산 공유메모리를 기반으로 각 시스템 노드에 작업량이 분배했을 때 작업소요시간을 비교하였다. <표 3>은 400회 연산을 수행하였을 때 각 시스템노드별 소요시간을 나타내고 있다.

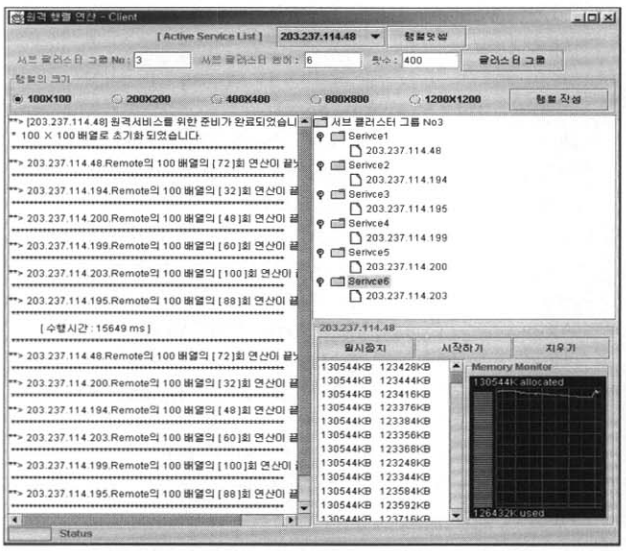
<표 3> 시스템노드별 작업소요시간

(단위 : ms)

System Node	IP	분산공유메모리	일괄 작업분류
1	203.237.114.48	3219	3141
2	203.237.114.194	2814	7294
3	203.237.114.195	2957	6314
4	203.237.114.199	3415	4016
5	203.237.114.200	2578	3060
6	203.237.114.203	2745	2914

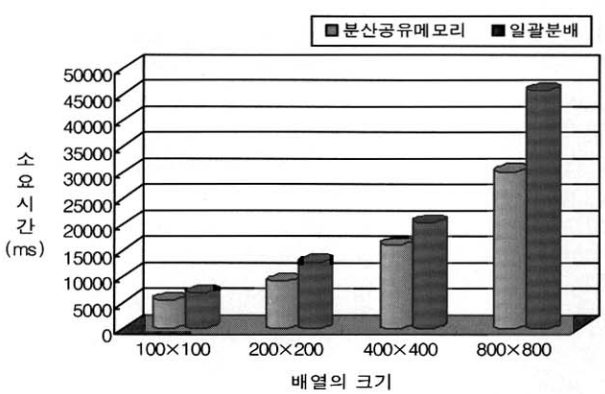
분산공유메모리를 기반으로 작업량을 분배하는 경우 전체 작업소요시간에 대한 표준편차가 312.3로 일괄분배방식 1884.5에 비하여 시스템별 편차를 최소화 시킬 수 있어서 전체적인 효율을 올릴 수 있음을 확인하였다. 따라서 전체 서브 클러스

터 그룹은 분산 공유메모리를 기반의 작업 분배를 통하여 전체적인 클러스터 그룹의 성능을 극대화시키고자 한다.



(그림 9) 100×100 행렬의 병렬 연산수행

(그림 9)는 100×100의 행렬 연산을 400회 수행할 때 서브 클러스터 그룹을 구성하고 있는 시스템 노드에 각각 분산 작업을 요청하고 있는 SC-Server의 실행모습이다. 서브 클러스터 그룹 No.3을 구성하고 있는 시스템 노드들은 요구된 작업의 양을 분산 공유메모리를 통하여 수행 가능한 작업의 양만큼 분배받는다. 각각의 작업대상 행렬을 실험한 결과 (그림 10)과 같은 작업소요시간을 얻었다.

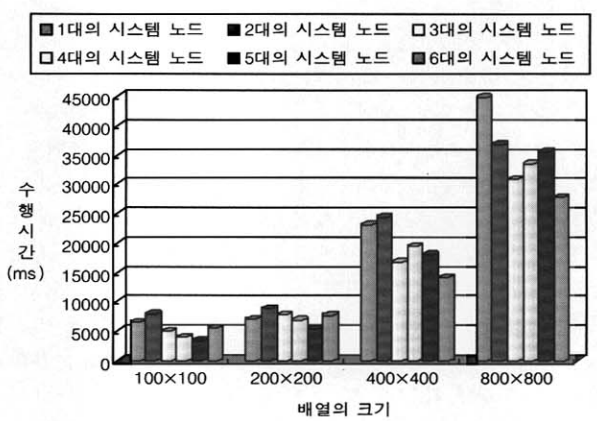


(그림 10) 작업행렬의 400회 연산

비교적 작은 작업량에 해당되는 100×100 행렬의 경우 분산공유메모리를 기반으로 한 작업분배 방식을 적용하여 연산을 수행한 경우, 일괄적인 작업분배방식과 비교하여 거의 비슷한 소요시간을 보였다. 그러나 작업량이 많아질수록 수행 소요시간의 차이는 더 벌어져 가장 큰 800×800 배열의 경우 51.8%의 소요시간 감축을 볼 수 있었다. 따라서 많은 연산이 소요되는 대규모 연산의 경우 시스템 노드의 성능에 따른 분산공유메모리기반의 작업분배방식을 이용하면 더욱 큰 시스템의 성능 향상을 볼 수 있음을 실험을 통해서 확인하였다.

4.2.2 시스템 노드 수에 따른 수행시간

이 실험에서는 시스템 노드의 수에 따라 소요되는 전체 연산시간을 측정하였다. 일반적으로 많은 시스템 노드들이 작업에 참여하는 경우 이에 비례하여 작업 소요시간이 감소할 것으로 예측할 수 있다. 그러나 개방 네트워크 상에 분산되어있는 시스템 노드의 특성상 대규모 자료를 전송하기 위한 네트워크 트래픽이 증가함에 따라 전체적인 연산시간이 증가할 수 있다. 따라서 본 실험에서는 요청되는 작업의 크기에 따라서 가장 최적의 서브 클러스터 그룹의 시스템 노드 수를 측정하고자 한다. 먼저 각각의 행렬에 대하여 시스템 노드 수를 달리하면서 400회의 연산을 수행하였다. 수행결과는 (그림 11)과 같다.



(그림 11) 시스템 노드 수에 따른 작업의 소요시간

(그림 11)의 시스템 노드 수에 따른 작업소요시간 그래프를 보면 100×100과 200×200 행렬의 연산소요시간은 서브 클러스터 그룹을 구성하고 있는 시스템 노드의 수가 5대일 때 가장 빠른 연산시간을 나타내고 있으며, 400×400과 800×800 배열의 경우 시스템 노드의 수가 6대일 때 가장 효율적임을 나타내고 있다. 시스템 노드가 1대인 경우는 로컬 시스템에서의 연산수행을 의미한다. 전체 행렬에 대하여 시스템 노드의 수가 1대인 경우에 비하여 2대에서 수행한 결과가 작업시간이 약간 더 소요됨을 보였다. 또한 100×100과 200×200 행렬의 경우 시스템 노드가 3대 이상 증가할수록 소요시간이 점점 단축되다가 5대의 시스템 노드일 때 최적의 성능을 보였고, 400×400과 800×800의 경우 6대의 시스템 노드일 때 최적의 성능을 보였다. 그러나 100×100과 200×200 행렬의 경우 6대 이상의 시스템 노드가 작업에 참여하는 경우 오히려 작업의 소요시간이 증가함을 볼 수 있었다. 이는 작업 데이터의 전달을 위한 소요되는

전송시간이 전체적인 연산소요시간에 비하여 상대적으로 증가하기 때문이다. 연산작업에 참여하는 시스템 노드의 수가 증가할수록 각 시스템에 할당되는 연산의 양은 줄어들어 그만큼의 연산소요시간의 단축효과를 볼 수 있지만 이와 비례하여 각 시스템 노드에 전송해야 할 데이터의 양 또한 증가되게 된다. 따라서 많은 데이터를 일시에 각 시스템 노드에 전송해야하므로 네트워크의 부하에 의한 자료전송시간의 증가가 전체 서브 클러스터 그룹의 성능을 저하시키는 요인이라 볼 수 있다. 400×400과 800×800의 경우에도 이러한 현상을 볼 수 있었다. 최적의 시스템 노드 수 이상의 시스템이 작업에 참여하는 경우, 각 시스템 노드에서 소요되는 연산시간은 그만큼 줄어들지만 데이터를 전송시간이 늘어남에 따라 전체작업의 완료시간은 오히려 증대되므로 네트워크 환경의 개선과 연산 대상이 되는 데이터 양에 따른 시스템 노드의 수가 동적으로 구성된다면 전체적인 작업의 효율은 더욱 향상될 수 있다.

5 결 론

제안한 다중 분산 웹 클러스터 모델은 기존의 클러스터 시스템을 기반으로 하여, 주어진 작업에 대한 병렬수행 및 SC-Sever의 공유메모리를 통한 효율적인 작업 분배와 시스템 노드간의 상호협조 작업을 통하여 고성능, 고효율 그리고 고가용성을 얻을 수 있는 웹 클러스터 모델이다.

서브 클러스터 그룹의 SC-Server는 가상서버의 작업요구를 수행하기 위하여 전체 시스템 노드의 서비스 코드 블록을 저장하고 있는 Lookup Server에 서비스 수행 코드를 요구하며, 전달받은 서비스 코드 블록을 중심으로, 분석된 작업의 내용이 각 시스템 노드에서 수행된다. 작업의 진행은 각 시스템 노드의 Distributed Block 모듈에 의해서 병렬 수행되어지며, 수행결과는 SC-Server의 Job Distributed Module에 모아져 사용자에게 전송하게 된다. 이를 위하여 SC-Server는 시스템 노드들과 Job Distributed Module의 분산 공유 메모리를 기반으로 한 자료 전달 방식을 이용한다.

향후 연구과제로는 대용량의 연산 작업 시 주어진 네트워크 환경에서 각 시스템 노드간의 효율적인 작업분배를 위한 고속의 데이터 전송방식 연구와 함께 분산된 시스템 노드간의 협조작업과 다양한 질의와 응답을 효율적으로 수행하기 위한 프로토콜의 표준화의 연구가 수행되어져야 하리라 사료된다.

참 고 문 헌

[1] R. Buyya, "High Performance Cluster Computing : Architectures and Systems," Prentice Hall, New Jersey, USA. Vol.1, 1999.

[2] A. Beguelin, J. Dongarra, A. Geist, R. Manch and V. Sunderam, "A User's Guide to PVM Parallel Virtual Machine," Technical Report CS-91-136, University of Tennessee, Jul., 1991.

[3] G. Burns, R. Daoud and J. Vaigl, "LAM : An Open Clutster Environment for MPI," Technical Report, Ohio Supercomputer Center, 1994.

[4] D. Andresen, T. Yang and O. H. Ibarra, "Towards a Scalable Distributed WWW Server on Workstation Clusters," Proc. of the 10th IEEE Intl. Symp. of Parallel Processing (IPPS'96), pp.850-856, April, 1996, http://ww w.csucsb.edu/Research/rapid_sweb/SWEB.html.

[5] H. J. Siegel, S. Abraham, W. L. Bain, K. E. Batcher, T. L. Casavant, "Report of the Purdue Workshop on Grand Challengers in Computer Architecture for the Support of High Performance Computing," Journal of Parallel and Distributed Computing 16, pp.199-221, Nov., 1992.

[6] P. Carns, W. Ligon III, R. Ross and R. Thakur, "PVFS : A parallel file system for linux clusters," In Proc. of the 4th Annual Linux Showcase and Conf, 2000.

[7] W. Zhang, S. Jin and Q. Wu, "Creating Linux Virtual Servers," LinuxExpo, 1999.

[8] S. Engelshall, "Load Balancing Your Web Site : Practical Approaches for Distributing HTTP Traffic," Web Techniques Magazine, Vol.3, No.5, May, 1998, http://www.webtechniques.com.

[9] R. Samanta, A. Bilas, L. Iftode and J. Singh, "Home-based SVM Protocols for SMP Clusters : Design and Performance," In Proc. of the 4th International Symposium on High-Performance Computer Architecture, pp.113-124, 1998.

[10] D. J. Becker, T. Sterling, D. Savarese, J. E. Dorband, U. A. Ranawak, C. V. Packer, "Beowulf : A Parallel Workstation for Scientific Computation," Proceedings of International Conference on Parallel Processing, 1995.

[11] C. Yoshikwa, B. Chun, P. Eastharn, A. Vahdat, T. Anderson and D. Culler, "Using Smart Cllient to Build Scalable Services," USENIX'97, 1997.

[12] T. Kwan, E. McGrath and A, Reed, "NCSA's World Wild Web Server : Design and Performance," IEEE Computer, pp.68-74, November, 1995.

[13] R. Nelson, "Exploring High Availability Issues with BEA Tuxedo and Third Party High Availability Software," Aurora information Systems, January, 1998.

이 기 준

e-mail : leekj@www.kjhc.ac.kr

1994년 조선대학교 전산통계학과(이학사)

1997년 조선대학교 일반대학원 전산통계학과(이학석사)

2001년 조선대학교 일반대학원 전산통계학과(이학박사)

2002년~현재 광주보건대학 컴퓨터보안과 전임강사

관심분야 : 클러스터 시스템, 분산 에이전트, 인공지능, 네트워크 보안