생태계 모방 시스템을 위한 OMNeT++기반 병렬 시뮬레이터의 설계 및 PC 클러스터 상에서의 성능 분석

(Design of an OMNeT++ based Parallel Simulator for a Bio-Inspired System and Its Performance on PC-Clusters)

문 주 선 † 낭 종 호 ††
(Joosun Moon) (Jongho Nang)

요 약 생태계 모방형 시스템[1]은 생태계에서 여러 객체들의 진화 및 협동 과정을 모방한 계산 모델로써, 기존의 알고리즘으로는 해결하기 어려운 문제들을 해결할 수 있는 방법으로 많은 연구가 진행되고 있다. 하지만, 이런 시스템은 많은 수의 객체가 진화 및 협동을 하는 과정을 필요로 하기 때문에 이런 시스템에 바탕을 둔 응용 시스템을 설계/분석하는데 많은 시간을 필요로 한다는 문제점을 가지고 있다. 본 논문에서는 이런 생태계 모방형 시스템의 동작을 확인할 수 있는 시뮬레이터를 여러 대의 PC상에서 동작하는 OMNeT++[2]를 확장하여 설계/구현하고, 몇 가지 응용 시뮬레이션을 통하여 그 유용성을 증명한다. 제안한 병렬 시뮬레이터에서는 Ecogent라는 객체가 진화/협동할 수 있는 기능을 제공하는 ERS 플랫폼을 OMNeT++에서 제공하는 기능으로 사상하여 여러 개의 플랫폼 상에서의 Ecogent가 동시에 진화/협동할 수 있게 함으로써 시뮬레이션 시간을 단축시킨다. 시뮬레이션 과정과 결과는 시뮬레이션 모니터 GUI를 통해서 실시간으로 확인할 수 있으며, 또한 시뮬레이션 결과의 체계적인 관리를 위하여 각 시뮬레이션 결과는 데이타베이스를 통해 저장되고 관리된다. 본 논문에서는 4개의 PC로 이루어진 PC cluster상에서 다양한 응용에 대한 생태계 모방형 시스템의 시뮬레이션 및 분석을 통하여 그 유용성을 검증하였다.

키워드 : 생태계, 생태계 모방형 시스템, 에코전트, 병렬 시뮬레이터

Abstract The Bio-Inspired system is a computing model that emulates the objects in ecosystem which are evolving themselves and cooperate each other to perform some tasks. Since it could be used to solved the complex problems that have been very difficult to resolve with previous algorithms, there have been a lot of researches to develop an application based on the Bio-Inspired system. However, since this computing model requires the process of evolving and cooperating with a lot of objects and this process takes a lot of times, it has been very hard to develop an application based on this computing model. This paper presents a parallel simulator for a Bio-Inspired system that is designed and implemented with OMNeT++ on PC clusters, and proves its usefulness by showing its simulation performance for a couple of applications. In the proposed parallel simulator, the functions required in the ERS platform for evolving and cooperating between objects (called Ecogent) are mapped onto the functions of OMNeT++, and they are simulated on PC clusters simultaneously to reduce the total simulation time. The simulation results could be monitored with a GUI in realtime, and they are also recorded into DBMS for systematic analyses afterward. This paper shows the usefulness of the proposed system by analyzing its performances for simulating various applications based on Bio-Inspired system on PC clusters with 4 PCs.

Key words : Bio, Bio-Inspired System, Ecogent, Parallel Simulator

† 학생회원 : 서강대학교 컴퓨터공학과
js2003@sogang.ac.kr
†† 종신회원 : 서강대학교 컴퓨터공학과 교수
jhnang@sogang.ac.kr
논문접수 : 2007년 6월 13일
심사완료 : 2007년 7월 22일

1. 서 론

급속한 인터넷의 발전으로 인해 앞으로 예견되는 응용 서비스들은, 전 세계 수많은 네트워크의 노드들을 하나로 연결하는 네트워크 기반 응용 서비스들이 주류를

이룰 것으로 예상된다. 이러한 대규모 시스템에서 동작하는 응용/시스템 소프트웨어에 대한 해결책으로 생태계 모방 계산모델[1]이 등장하게 되었다. 이러한 연구들 중에서 에코전트(Ecogent, Ecology와 Agent의 합성어)기반의 생태계 모방형 시스템이 다른 네트워크 기반 시스템과 구별되는 가장 큰 특징은 확장성, 적응성, 생존성을 갖는 네트워크를 구성할 수 있다는 것이다. 확장성은 양적으로 매우 크게 증가하는 호스트의 수를 감당할 수 있음을 의미하고, 적응성은 다양한 환경 변화에 네트워크가 스스로 적응함을 의미한다. 생존성은 많은 호스트가 복잡하게 연결되어 있는 네트워크에서 몇 개의 호스트가 결함이 있어도 전체 서비스에 영향이 없어야 함을 의미한다.

이러한 특성을 검증하기 위해서는 대단위 호스트를 가진 네트워크의 구성이 필수적이다. 하지만 시스템이 개발 단계이거나 새로운 응용 및 네트워크 토폴로지를 실험하고자 할 때마다 수천수만 개의 실제 호스트들을 연결한다면 매우 큰 비용을 필요로 할 것이다. 따라서 대단위 호스트들을 자유롭게 네트워크화하고 새로운 응용의 이식이 쉬운 효과적인 시뮬레이터의 개발은 전체 시스템과 응용의 기능 및 성능 검증을 위해 매우 중요한 부분이라고 할 수 있다.

본 논문에서는 생태계 모방형 시스템을 시뮬레이션 할 수 있는 시뮬레이터를 설계 및 개발한다. 시뮬레이션 시스템은 바이오 플랫폼 시뮬레이터, 데이타베이스, 그리고 GUI 모니터로 구성된다. 시뮬레이터는 객체 지향 이벤트 기반 시뮬레이터 엔진인 OMNeT++[2]를 기반으로 여러 응용의 이식 및 플랫폼 수정을 용이하게 하기 위해 계층적 방법을 사용하여 설계한다. 또한 MPI 기반의 병렬 수행을 통해 시뮬레이션 가능한 호스트 수의 증가와 시뮬레이션 속도를 향상시킨다. 시뮬레이션 결과는 데이타베이스화하여 체계적으로 관리되고, 언제든지 GUI 기반의 시뮬레이션 모니터를 사용하여 지난 시뮬레이션 결과를 확인할 수 있다. 시뮬레이션 모니터는 현재 수행하고자 하는 시뮬레이션의 환경 설정 및 제어 기능을 제공하고, 시뮬레이션 진행 상황과 결과를 애니메이션이나 그래프로 보여준다. GUI 모니터는 각 응용에 적합한 결과 화면을 쉽게 삽입할 수 있도록 컴포넌트 구조로 설계 및 구현되었다. 개발된 시뮬레이터는 서비스 시나리오에 의한 서비스 단위의 기능 검증과 소규모 네트워크에서의 실험 결과 비교를 통한 시스템 단위 기능 검증 과정을 거쳐 올바른 시뮬레이션 결과를 만들어냄을 확인하였다. 또한 병렬 수행 실험을 통해 대단위 노드로 네트워크를 구성했을 때 실제 시뮬레이션의 속도가 향상됨을 확인하였다.

2. 관련연구

2.1 생태계 모방형 시스템

생물학자들의 개미 군집에 대한 연구 이후, 생태계 모방에 대한 연구[4,5]는 학계에서 비상한 관심을 갖게 되었다. 개미의 유전자나 뇌 조직에는 계획이나 조직성, 제어 기능이 존재하지 않지만, 자발적인 조직화 과정을 통하여 문제를 해결해 나간다. 이러한 생태계의 특징을 모방하여 대규모 네트워크 상에서의 시스템 개발에 생태계적 접근을 시도한 것이 생태계 모방형 시스템인데, 이 시스템의 대표적인 모델로 UCI에서 개발한 'Bionet Platform'[6]이 있다. 이 플랫폼은 Cyber Entity라 불리는 작은 개체들과 이 개체들 간의 단순하고 다양한 행동 및 각 개체 간의 자율적인 상호작용에 의해 동작한다.

하지만, 이 플랫폼은 생태계 모방형 시스템의 주요 특징 중 하나인 생존성을 지원하지 못한다. 대규모 시스템에서 동작하는 소프트웨어에서는 작은 시스템 오류 하나라도 전체 시스템에 대한 치명적인 위험을 끼칠 수 있으므로, 시스템의 자동 오류 복구 능력은 매우 중요하다고 볼 수 있다. Bionet Platform에서 제공하는 서비스들은 확장성과 적응성의 기능은 지원하지만, 이러한 생존성을 해결하기 위한 서비스가 없기 때문에, 대규모 네트워크를 위한 안정적인 시스템의 구축에는 부족한 점이 많다. 생태계 모방 플랫폼은 Fault Tolerance 기능을 제공함으로써 생존성 문제를 해결하였으며, 더불어 응용 시스템 개발에 이 플랫폼의 사용 및 이식이 편리하도록 구현하였다.

2.2 OMNeT++

OMNeT++[2]는 1992년에 Budapest 공과대학에서 개발한 객체 지향 이벤트 기반 시뮬레이터 엔진이며 통신 프로토콜, 컴퓨터 네트워크, 멀티프로세서, 그리고 분산 시스템 등 다양한 형태의 시뮬레이션을 위해 사용할 수 있다. OMNeT++의 시뮬레이터 커널은 네트워크 구성, 메시지 전달, 이벤트 관리 등 시뮬레이션에 필요한 기본 기능들만을 제공함으로써 에이전트 시스템과 같은 네트워크 시뮬레이션이 아닌 것을 이식하기에 용이하며, 불필요한 기능이 제거된 작은 크기의 커널은 실행 시 큰 효율성을 제공한다. 또한 매우 큰 크기의 시뮬레이션을 빠른 시간에 수행할 수 있도록 MPI를 기반으로 한 병렬 수행을 지원한다. 무엇보다 15년의 역사와 풍부한 사용 예를 통해서 견고하고 우수한 시뮬레이터 엔진임을 알 수 있다.

2.3 ERS (Ecogent Runtime Services) Model

생태계 모방형 시스템은 확장성, 적응성, 생존성을 가져야 한다. 이 특징을 지원하기 위해 생태계 모방형 시스템은 Registration, Life Cycle, Communication, Location, Migration, Fault Tolerance와 같은 Ecogent

Runtime Service들을 제공한다. 각각의 서비스들에 대한 설명은 다음과 같다.

- Registration: 이 서비스는 에코전트 객체들을 해당 플랫폼과 endpoint에 등록하고 endpoint로부터 고유한 ID를 생성 받는 작업을 담당하는 서비스로서 다른 서비스들의 작업을 위한 에코전트의 reference 및 에코전트에 관한 정보를 제공하게 된다.
- Life Cycle: 에코전트를 사용하는 상위 계층에서 에코전트를 생성, 소멸시키고, 각각의 목적에 맞게 에코전트의 상태를 변화시키기 위한 서비스이다.
- Migration: 에코전트의 요청에 따라 코드와 데이타를 다음에 실행할 다른 플랫폼으로 이동시킨다.
- Communication: 에코전트와 에코전트간의 메시지를 전달하는 서비스로, ACL Message를 사용하며, 내부적인 통신은 ERS의 하위 계층인 endpoint를 이용한다.
- Location: Location은 다른 ERS 플랫폼을 찾는 서비스로, 지역 플랫폼의 Registration 서비스를 이용하여 원하는 정보를 찾게 된다.
- Fault Tolerance: Fault-Tolerant 기능을 수행하기 위해, 에코전트의 상태를 주기적으로 기록하거나 에코전트의 복사본을 동시에 수행하는 서비스이다.

3. 생태계 모방형 시스템 시뮬레이터 구조

그림 1과 같이 생태계 모방형 시스템 시뮬레이터는 OMNeT++ 시뮬레이터 엔진부터 바이오 플랫폼까지 계층적 방법(Layered Approach)을 사용하여 설계되었다. 시뮬레이터 엔진은 시뮬레이션 대상인 플랫폼과 직접 통신하지 않고 중간 계층인 플랫폼/커널 매핑 계층을 통해 통신하도록 설계함으로써 시뮬레이션 대상의 수정 및 변화가 용이하도록 하였다. 본 장에서는 각 계층에 대해 자세히 설명하도록 한다.

3.1 MPI를 이용한 OMNeT++의 병렬 시뮬레이션

그림 1에서 볼 수 있듯이, OMNeT++는 생태계 모방형 시스템의 시뮬레이션 작업을 병렬 처리하기 위해, MPI(Message Passing Interface)를 이용하여 여러 개의 PC들을 하나의 cluster로 묶은 후, 시뮬레이션을 수행한다. OMNeT++에서 이러한 병렬 시뮬레이션의 수행을 담당하는 곳은 '병렬 시뮬레이션 서브시스템'인데, 이 서브시스템은 아래와 같은 세 개의 층들로 구성되며, OMNeT++는 이 층들 간의 상호 동작으로 인해 MPI Cluster와의 통신이 가능하다.

- Communication Layer: 이 layer의 목적은 상위 layer의 partition에 메시지를 전달하는 것이며, send/blocking_receive/nonblocking_receive/broadcast의 4가지 서비스로 구성되어 있다.
- Partitioning Layer: 이 layer는 proxy gate를 구성하기 위해서, configuration에 맞게 서로 다른 Logical Process간의 모듈들을 만든다. 또한 시뮬레이션이 수행되는 동안, 각 시뮬레이션 메시지들이 원하는 목적지에 무사히 도착할 수 있도록 도와준다.
- Synchronization Layer: 이 layer는 병렬 시뮬레이션 알고리즘을 포함하고 있다. 병렬 시뮬레이션 알고리즘은 이벤트 스케줄링, 메시지 전송과 수신, 메시지간의 synchronization에 대한 기능을 제공한다.

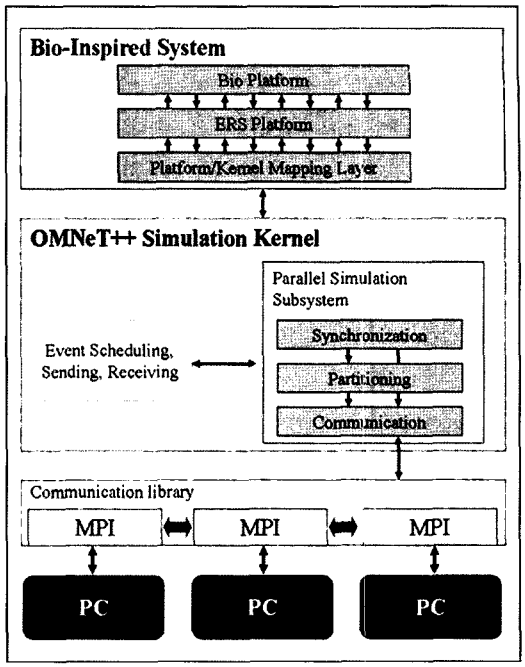


그림 1 생태계 모방형 시스템 시뮬레이터 구조

3.2 플랫폼/커널 매핑 계층

플랫폼/커널 매핑 계층은 시뮬레이션 하고자 하는 플랫폼의 이식 및 유지/보수의 편리성을 위하여 커널과 플랫폼의 중간 역할을 하는 핵심 계층으로 다음과 같이 3개의 주요 모델로 구성되어 있다.

- PlatformBase: 이것은 시뮬레이터 엔진이 제공하는 cSimpleModule이라는 클래스를 상속 받은 클래스이다. cSimpleModule은 시뮬레이터 엔진과 직접적으로 통신할 수 있는 클래스로서, 이벤트를 받아 처리하거나 다른 모듈과 네트워크를 형성할 수 있다. 따라서 PlatformBase는 각 바이오 플랫폼 모듈의 시뮬레이션 진행을 담당하며, 각 플랫폼이 처리해야 할 이벤트를 받는 통로 역할을 한다. 또한, ERS 플랫폼이 필요로 하는 서비스를 시뮬레이터 커널에 요청할 수 있어야 한다.
- AdminService: 시뮬레이터 커널과 직접적인 관련이 없지만, 각 플랫폼의 메모리나 CPU의 리소스를 가상

으로 관리하는 기능을 담당한다. 시뮬레이션 되고 있는 각 플랫폼들의 실제 메모리 사용량이나 CPU 점유량을 알아내는 것은 불가능하기에, 본 시뮬레이터에서는 다음과 같이 추측하여 계산하였다. 먼저 메모리 사용량은 플랫폼 클래스와 에코전트 클래스의 크기를 구하여 추측하였다. 동적으로 할당되고 해제되는 메모리까지 계산하여 실제 메모리 사용량에 근접할 수 있도록 하였다. CPU 점유율은 CPU 상태에 영향을 주는 요소가 매우 다양하고 변화가 심해서 정확히 추측하기 힘들지만, 각 플랫폼의 CPU에 영향을 주는 요소가 에코전트 활동 외에는 동일하다고 가정한다면 에코전트의 시간 복잡도를 사용하여 CPU 점유도를 추측할 수 있다. Software metric분야에서 주로 사용되는 클래스 복잡도 계산방법인 McCabe[3]방법을 사용하여 에코전트의 시간 복잡도를 계산하였다.

- DBManager: DBManager 모듈은 플랫폼 단의 시뮬레이션 결과를 받아 결과 DB에 저장하는 일을 수행한다. 결과 DB 접근은 반드시 DBManager 모듈을 통해서만 이루어진다.

3.3 ERS (Ecogent Runtime Services) 플랫폼과 바이오 플랫폼

ERS는 Registration, Life Cycle, Migration, Communication, Location, Fault tolerance 서비스들로 구성되어 있으며, 각 서비스를 설계된 API의 문법(syntax)과 의미(semantics)를 최대한 준수하여 C++로 재 구현 후 이식하였다. 또한, 각 서비스는 하위 계층인 플랫폼/커널 매핑 계층에서 제공해주는 인터페이스만을 사용하여 구현하였다.

바이오 플랫폼은 ERS 플랫폼의 서비스를 바탕으로 움직이도록 설계되었다. 따라서 시뮬레이터에서도 바이오 플랫폼은 ERS 플랫폼의 상위 계층에 존재하여 ERS 서비스를 이용하여 구성된다. 하지만, 응용 프로그램은 바이오 플랫폼과 ERS 플랫폼 서비스를 동시에 사용가능하도록 구성하였다.

계층 구조로 되어 있는 바이오 플랫폼 시뮬레이터를 사용하여 *n*개의 바이오 플랫폼을 시뮬레이션 한다고 가정하자. 시뮬레이터 엔진은 *n*개의 플랫폼을 시뮬레이션 하기 위해서 n개의 PlatformBase를 만들고 관리한다. 엔진은 시뮬레이션 시간을 관리하며 각 PlatformBase가 자체적으로 가지고 있는 Event Queue에서 event를 꺼내 처리할 수 있도록 순차적으로 실행 기회를 준다. 실행 기회를 받은 각 PlatformBase는 플랫폼을 활성화하고 현재 플랫폼에 등록되어 있는 각 에이전트들에게 실행 기회를 준다.

그림 2는 이벤트 처리과정을 그림으로 보여준다.

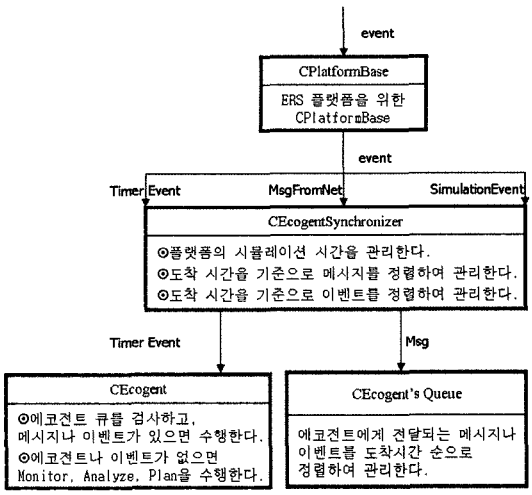


그림 2 플랫폼 이벤트 처리 과정

PlatformBase가 받은 이벤트는 EcogentSynchronizer에게 전달되어 '타이머 이벤트', 다른 플랫폼에서 전달된 '메시지 이벤트', 그리고 '시스템 이벤트'의 3종류로 구분되어 처리된다. 타이머 이벤트는 이벤트를 받은 플랫폼이 실행 기회가 되었다는 것을 의미하며 플랫폼은 자신에게 등록되어 있는 에코전트들에게 수행 기회를 준다. 실행 기회를 얻은 에코전트는 자신의 메시지 큐와 명령(command) 큐를 검사하여 메시지나 명령이 존재하면 그것을 처리하고, 존재하지 않는다면 에코전트의 Monitor, Analyze, Plan 과정을 차례로 수행한다. 다른 플랫폼에서 받은 메시지 이벤트는 EcogentSynchronizer가 도착시간 순서대로 각 에코전트에게 배달하며, 메시지를 받은 에코전트는 자신의 메시지 큐에 삽입하여 실행 기회가 왔을 때 꺼내어 처리한다. 마지막으로 시뮬레이션 이벤트는 플랫폼이 크래쉬(Crash) 되거나 시뮬레이션이 종료가 될 경우처럼 특별한 상황이 발생할 때 전해지는 이벤트이며 이 이벤트 역시 EcogentSynchronizer가 받아 메시지로 변환하여 각 에코전트에게 전달한다. 이 메시지 역시 에코전트의 메시지 큐에 저장되며 일반 메시지와 같은 방법으로 처리된다. 이렇게 n개의 플랫폼이 모두 실행 기회를 얻어 수행되고 나면 시뮬레이션 시간은 정해진 크기만큼 증가하고 플랫폼 실행 과정을 반복한다.

4. 시뮬레이션 결과 관리

본 시뮬레이션 시스템은 결과의 체계적인 관리를 위하여 DB를 통해 시뮬레이션 결과를 저장 및 관리한다. 표 1과 그림 3은 본 시뮬레이터에서 사용한 DB 스키마의 다이어그램과 주요 테이블이다.

표 1 주요 테이블 설명

Table 이름	설명
E_Simulation	시뮬레이션 ID, 제목 등 시뮬레이션의 기본 정보
E_TopologyModel	시뮬레이션에 사용한 네트워크 토폴로지 유형
E_TopologyPlatform	시뮬레이션 플랫폼 기본 정보
E_TopologyPlatform History	E_TopologyPlatform의 변화에 대한 history 정보
E_EcogentClass	에코전트의 종류에 대한 정보
E_EcogentRegistration	각 플랫폼별 에코전트의 등록 정보
E_EcogentRegistrationHistory	E_EcogentRegistration의 변화에 대한 history 정보
E_MPIClusterNode	시뮬레이션에 사용한 MPI 노드들의 시스템 정보
E_TopologyPlatform AppData	응용에서 서로 다른 목적으로 사용할 테이블
E_TopologyPlatform AppDataHistory	E_TopologyPlatformAppData의 변화에 대한 history 정보

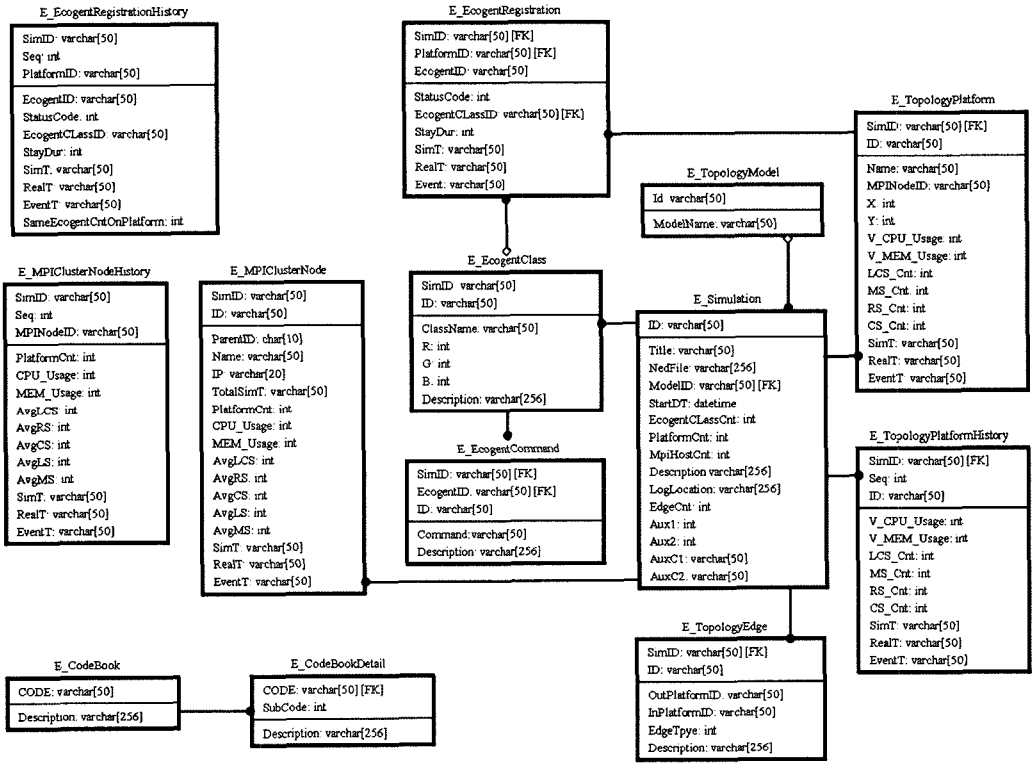


그림 3 바이오 플랫폼 시뮬레이터 DB 스키마 다이어그램

5. 시뮬레이터 시스템의 구현

그림 4와 같이 시뮬레이터 시스템은 시뮬레이터, 데이타베이스, 그리고 모니터의 3개 부분으로 구성되어 있다. 본 논문에서는 1.5 GHz CPU와 512 MB 메모리를 가진 6대의 컴퓨터를 사용하여 시스템을 구성하고 실험하였다. 이 중 4대는 Linux를 설치하고 MPI 클러스터로 구성하여 시뮬레이터를 탑재했고, 2대는 Windows XP를 설치하여 각각 MS-SQL 서버와 모니터로 구성하였다.

5.1 시뮬레이션 모니터

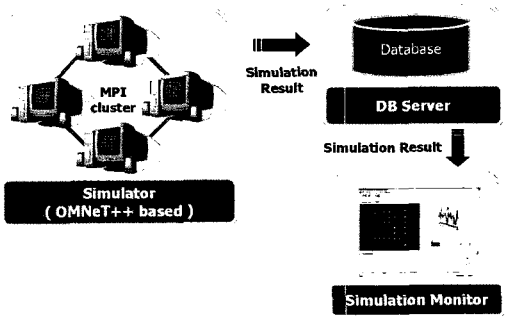


그림 4 시뮬레이터 시스템 구조

시뮬레이션 모니터는 시뮬레이션 된 과정 및 결과를 시각적으로 확인하기 위한 GUI 환경으로서 사용자는 모니터를 통해서 시뮬레이션의 과정 및 결과를 쉽게 비교할 수 있다. 시뮬레이션 모니터 설계의 초점은 시뮬레이션 응용 개발자가 보여주고 싶은 것을 자유롭게 표현할 수 있도록 하는 것이었다. 이러한 관점을 반영한 설계가 토폴로지 창에서 플랫폼 색상을 개발자가 직접 지정할 수 있게 한 것과 결과 분석 창의 결과 분석 도구를 OCX 형태로 개발해 끼워 넣을 수 있게 설계 한 것이다. 또한, 시뮬레이터 자체의 메모리 상태나 CPU 점유율 등의 성능 표현 및 각종 부가 정보 등을 표현함으로써 시뮬레이션 모니터를 통해서 시뮬레이션 결과의 모든 것을 확인할 수 있도록 설계 및 구현하였다. 그림 5는 5개의 창(windows)으로 구성되어 있는 시뮬레이션 모니터 GUI를 보여준다.

- 시뮬레이션 정보 창: 현재 데이타베이스에 저장되어 있는 시뮬레이션 리스트 정보를 보여주는 창이다. 시뮬레이션 ID, 제목, 시간, 초기화 정보 파일이름 등이 표현되며, 사용자가 리스팅 된 시뮬레이션 ID중 하나를 선택 시 해당 시뮬레이션이 모니터에 보인다.
- 토폴로지 창: 매우 많은 수의 플랫폼을 시뮬레이션 할 경우 2차원 평면에서 각 플랫폼을 효과적으로 표현하는 것은 불가능하다. 따라서 시뮬레이션 모니터는 3차원으로 토폴로지를 재구성하여 표현하며, 플랫폼의 색상은 응용 개발자가 매 순간 지정하여 입력할 수 있게 하였다. 그림 6은 900개의 플랫폼을 GRID 네트워크 구조로 표현한 것이다.
- 에코전트 정보 창: 그림 7에서 볼 수 있듯이, 현재 시뮬레이션에서 사용되고 있는 에코전트 정보를 보여준다. 에코전트의 이름, 역할, 개체 수 등이 표현된다.
- 시뮬레이션 모니터 제어 창: 시뮬레이션 모니터를 시작, 중지 및 모니터 속도까지 조절 가능한 제어 박스가 있다. 사용자는 모니터 제어 박스를 사용하여 시뮬레이션 진행을 조절할 수 있다.
- 결과 분석 창: 이 창에는 각 시뮬레이션 응용 개발자가 자신들의 응용을 가장 잘 표현할 수 있는 결과 분석 도구를 제작하여 표현한다. 결과 분석 도구는 OCX 형태로 제작되어 시뮬레이션 응용에 따라서 결과 분석 도구가 바뀔 경우 별도의 컴파일 과정 없이 바로 결과 분석 창에 넣을 수 있도록 설계 하였다.

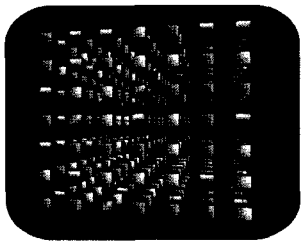


그림 6 토폴로지 창

SimID	ID	ClassName	R	G	B
1		CPoliceEcogent	0	0	255
1		CSuspectEcogent	255	0	0

그림 7 에코전트 정보 창

6. 시뮬레이터 시스템에 대한 실험

6.1 응용 서비스 별 시뮬레이터의 적용 실험

본 논문에서 제안한 생태계 모방형 시스템을 위한 병렬 시뮬레이터가 실제로 다양한 응용 프로그램 개발 시 어떻게 적용되고, 어떠한 편의성이 있는지 실제로 개발하여 실험해 보았다.

6.1.1 인터넷 서버의 과부하 분산 시스템

생태계 모방 플랫폼을 이용한 응용 프로그램으로 인터넷 교육방송용 서버의 과부하를 분산시키기 위한 load balancing system[7]을 개발하였다. 본 응용은 과부하가 걸린 서버에서 접속자를 idle상태의 서버로 보내고

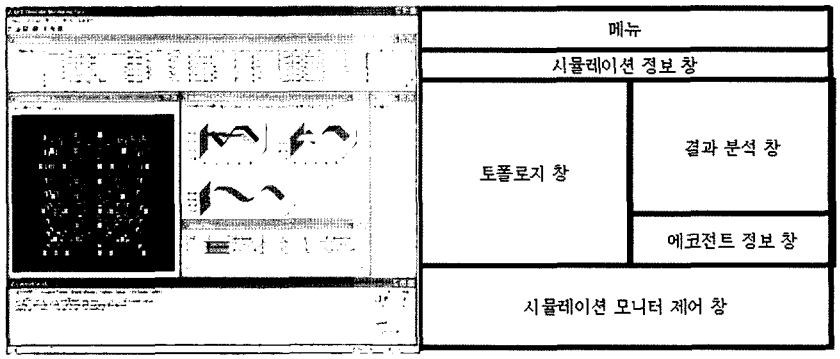


그림 5 시뮬레이션 모니터 GUI

idle상태의 서버에서는 과부하가 걸린 서버에서 접속자를 끌어오는 방식으로, 서버의 과부하를 막고 전체적인 서버의 활용도를 높이는 응용이다.

이러한 문제는 생태계 모방 플랫폼의 Stigmergy Control을 사용한다면 쉽게 해결할 수 있다. Stigmergy control에서는 각 에코전트가 다음 홉의 대상이 될 노드를 결정할 때, AntNet과 같은 개념의 라우팅 테이블을 유지한다. 각 router에는 ERS가 있고, 이런 라우터 상에서 Ecogent가 이동하며 주위 라우터에 대한 정보를 남긴다. 어떤 Ecogent가 자신이 지나간 노드에 대한 네트워크 부하 정보를 페로몬 형태로 해당 라우팅 테이블에 남기면, 과부하 발생시 이를 분산시키기 위해 이동해야 할 노드의 방향을 빠르게 제공하여, 보다 효율적이고 분산된 환경에 적합한 과부하 분산을 촉진하는 역할을 한다.

그림 8은 10×10 네트워크에서의 과부하 분산에 대한 표준 편차를 보여주고 있다. 그래프에서 보는 바와 같이, 생태계 모방 플랫폼을 이용한 Eco-LB는 최적으로 수렴한다고 증명되어 있는 순차적 확산 알고리즘[8]을 이용한 Diffusion 보다 더 나은 과부하 분산 수행능력을 보인다. 따라서 load balancing과 같은 응용 프로그램은 라우터에 ERS 플랫폼을 설치하고, BIO 플랫폼의 Stigmergy Control 기능을 이용하여 네트워크 부하 정보를 페로몬 형태로 남기는 방식으로 쉽게 개발할 수 있다.

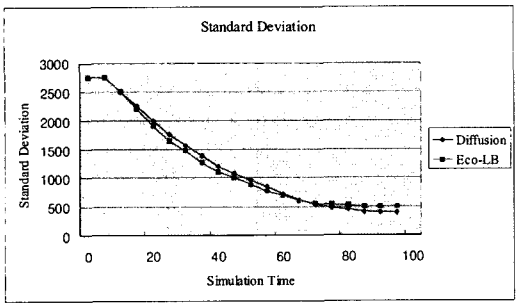


그림 8 네트워크에서의 과부하 분산 (표준 편차)

6.1.2 마이크로어레이 데이타 분석 시스템

마이크로어레이[9]의 개발로 각 연구기관, 병원에서는 많은 양의 마이크로어레이 데이타를 보유하게 되었고, 이들 데이타의 분석으로 각 종 질병에 대한 유전자의 반응과 환자의 병인 분석 및 치유책의 발견도 가능하게 되었다. 하지만 높은 단가의 마이크로어레이와 실험대상자 수의 부족으로, 더욱 정확한 마이크로어레이 데이타의 분석을 위해서는 다른 연구기관의 데이타들을 취합하여 분석해야 하는데, 문제는 이 데이타 분석이 매우 오래 걸릴 수 있다는 것이다.

이러한 문제점을 보완하기 위해 생태계 모방 플랫폼을 이용하여 해결할 수 있다. 이 시스템은 데이타를 직접 이동하지 않고 각 지역 노드에서 지역 데이타를 바탕으로 분석한 결과를 다른 노드의 지역 데이타와 같이 분석 가능한 대표성을 가진 형태의 정보로 압축하여 이웃 노드로 정보를 전달함으로써 효율적이고 보안상의 문제도 해결하는 장점을 가지고 있다.

또한 마이크로어레이의 특성상 자질(Feature) 종류의 수가 데이타의 수에 비해 굉장히 많다. 이러한 형태의 데이타의 경우 컴퓨터학습을 통한 데이타의 분석 결과가 training data에 전적으로 의존하게 되어 좋은 분석 결과를 주지 못하게 된다. 따라서 데이타의 분류에 가장 중요한 영향을 미치는 자질을 찾아내어야 비로소 정확한 마이크로어레이 데이타를 분석할 수 있게 된다. 이러한 자질을 찾아내는 과정을 Feature Selection이라고 하며 많은 방법들이 제시되었다. 우리는 생태계 모방 플랫폼을 이용한 유전자 알고리즘 기반의 Feature Selection을 통해 보다 효율적이고 압도적으로 적은 수의 자질을 추려냄으로써 데이타의 분석 정확도는 기존의 시스템에서 나온 정확도와 비슷하거나 상회하는 결과를 보였다.

6.2 대단위 플랫폼 네트워크 응용에 대한 시뮬레이션 속도 실험

본 시스템은 시뮬레이션 속도 향상을 위해서 다음과 같은 3가지 방법을 통해 구현하였다.

첫 번째로 시뮬레이터와 GUI를 포함한 시뮬레이션 모니터를 분리하였다. 만일 시뮬레이터와 함께 시뮬레이션 과정을 보여주는 GUI가 함께 수행된다면 시뮬레이션 시간은 매우 크게 늘어날 것이다. 수천수만 개 플랫폼의 변화 과정을 일일이 해석하고, 각 플랫폼마다 에이전트들의 이동 상황을 추적하여 화면에 그려주는 것은 매우 많은 시간을 소비하는 일이다. 본 시뮬레이터 시스템은 시뮬레이터가 데이타베이스를 사용하여 결과 데이타를 모니터에게 전달하기 때문에 모니터와 같은 컴퓨터에서 실행될 필요가 없다.

두 번째로 일괄 처리(Batch Process)를 사용하여 데이타베이스 접근 횟수를 줄였다. 시뮬레이션 시간 증가의 가장 큰 원인은 시뮬레이터가 시뮬레이션 수행 중에 데이타베이스를 매우 자주 액세스하며, 과정 모니터링을 위해서 정보의 변화가 생길 때마다 history 정보를 만들어야 한다는 점이다. 이러한 문제점을 해결하기 위해서 본 시뮬레이터에서는 일괄 처리를 사용하여 데이타베이스 갱신을 이원화 하였다. 즉, 시뮬레이션 수행 중에는 실시간 모니터링을 위한 최소한의 정보만을 데이타베이스에 기록하고, 나머지 정보와 history 정보는 파일로

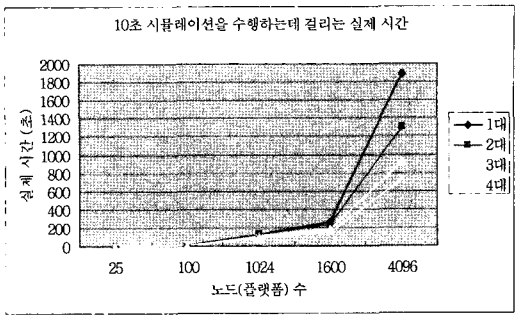
내려 시뮬레이션 종료 후에 일괄 처리로 데이타베이스에 삽입한다. 이러한 방법을 통해 시뮬레이션 속도가 약 60배 이상 향상되는 결과를 볼 수 있었다.

마지막으로 MPI를 사용한 병렬 수행을 통해 시뮬레이션 속도를 향상하였다. 전술한 두 개의 문제점을 해결했음에도 여전히 시뮬레이션 속도가 느린 것은 플랫폼 자체의 계산량이 많고 시뮬레이션 해야 할 플랫폼의 수가 매우 크기 때문이다. 따라서 시뮬레이션을 수행하는데 필요한 I/O, CPU 계산과 같은 다양한 부하를 여러 컴퓨터에서 나누어 수행하는 것이 최선의 해결책이 될 수 있다. 본 논문에서 시뮬레이션 엔진으로 사용한 OMNeT++는 MPI를 통한 병렬화가 가능하며 그 위에 탑재된 프로그램들 또한 병렬 수행을 고려하여 노드 간/모듈 간 의존도를 최대한 작게 하여 구현되었다.

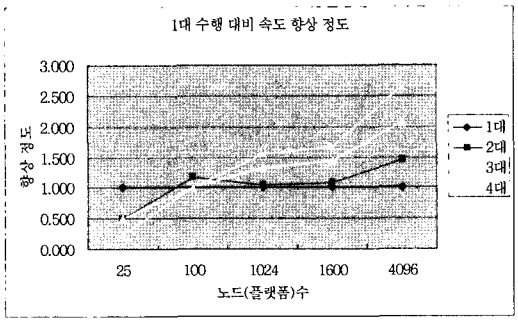
병렬 수행을 통한 속도 향상 정도를 실험적으로 검증하기 위하여 다음과 같은 실험을 하였다. 25, 100, 1024, 1600, 그리고 4096 개 플랫폼들이 연결되어 GRID 네트워크를 형성하고, 각 플랫폼에는 2개의 에코전트를 생성해 임의로 통신(communication)및 이주(migration)를 하도록 하였다. 이러한 조건으로 10초간의 시뮬레이션을 하기 위해서 실제로 소요되는 시간을 컴퓨터 1대에서 수행할 때와 여러 대의 컴퓨터에서 병렬 수행할 때를 나누어 비교 측정하였다.

그림 9(a)에서 보여주는 것처럼 시뮬레이션 대상이 되는 플랫폼 수가 늘어날수록 시뮬레이션에 소비 되는 시간이 큰 폭으로 늘어나는 것을 확인할 수 있다. 또한 병렬 수행에 사용한 컴퓨터 수가 많을수록 시뮬레이션 시간이 큰 폭으로 줄어드는 것으로 보아 병렬 수행이 시뮬레이션 속도 향상에 큰 역할을 하는 것으로 해석할 수 있다. 그림 9(b)는 병렬 수행을 하지 않았을 때의 시뮬레이션 속도를 1로 봤을 때 병렬 수행 시 나타나는 속도 변화를 보여준다. 실험 결과를 보면, 플랫폼 수가 작을 경우에는 오히려 병렬 수행을 했을 때 속도가 떨어지는 것을 볼 수 있다. 이는 병렬 수행을 위해서 각 컴퓨터가 통신하는데 사용하는 비용이 시뮬레이션 계산 시간보다 크기 때문에 발생하는 현상으로 해석할 수 있다. 하지만 플랫폼 수가 늘어남에 따라 속도 향상 정도가 크게 늘어나는 것을 확인할 수 있다.

컴퓨터 한 대당 시뮬레이션 가능한 플랫폼의 개수는 컴퓨터의 메모리 크기에 따라 달라진다. 플랫폼마다 각 서비스 오브젝트와 에코전트 오브젝트를 가지고 있기 때문에 플랫폼 당 요구하는 메모리가 작지 않기 때문이다. 실험에 의하면 2 GB의 메모리를 가지고 있는 컴퓨터에서 약 5,000개의 플랫폼을 시뮬레이션 했을 경우 95% 정도의 메모리 점유율을 보였다(그림 10). 5,000개 이상일 경우 메모리 점유율이 95~6%에서 수렴하는 것



(a) 10초 시뮬레이션을 위해 소요되는 시간



(b) 1대 수행 시간 대비 속도 향상 정도

그림 9 클러스터 구성 노드 수 확장에 대한 시뮬레이션 속도 변화

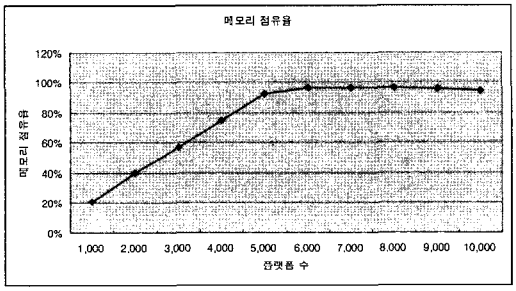


그림 10 플랫폼 개수에 따른 메모리 점유율 변화

으로 보아 가상 메모리가 작동 되는 것으로 볼 수 있으며, 이는 시뮬레이션 속도 저하의 큰 원인이 될 수 있다. 따라서 2GB 메모리일 경우 각 컴퓨터에서 시뮬레이션 할 수 있는 적정 플랫폼 수는 5,000개 정도라고 할 수 있다.

7. 결 론

본 논문에서는 생태계 모방형 시스템을 위한 시뮬레이터를 설계 및 구현하였으며 다양한 검증 방법을 사용하여 그 유용성을 확인하였다. 제안한 시뮬레이터는 OMNeT++를 기반으로 한 시뮬레이터 엔진 계층부터

응용 계층까지 4단계의 계층적 구조로 설계함으로써 플랫폼의 수정이나 응용의 이식이 수월하도록 설계하였다. 또한 MPI를 통한 병렬 시뮬레이션은 생태계 모방형 시스템의 가장 큰 특징인 확장성, 적응성, 생존성을 검증할 수 있도록 대단위 노드 수를 가진 네트워크 구성을 가능케 하고 시뮬레이션 속도 향상 효과도 가져올 수 있었다. 시뮬레이션 결과는 데이타베이스화하여 저장되고 GUI 모니터를 사용하여 재생할 수 있도록 하여 체계적으로 관리될 수 있도록 하였다. 본 논문에서 제안한 시뮬레이터는 생태계 모방형 시스템을 이용한 여러 응용 시스템들의 대단위 네트워크에서의 기능 및 성능 검증에 유용하게 사용할 수 있을 것이다.

참 고 문 헌

[1] M. Wang and T. Suda, "The Bio-Networking Architecture: A Biologically Inspired Approach to the Design of Scalable, Adaptive, and Survivable/Available Network Application," Proc. of the IEEE Symposium on Application and the Internet, 2001.

[2] A. Varga, "The OMNeT++ Discrete Event Simulation System," Proc. of the European Simulation Multiconference, 2001.

[3] T. McCabe and C. Butler, "Design Complexity and Measurement Testing," Communication of the ACM, Vol.32, No.12, 1989.

[4] D. Meslati, L. Souici, S. Ghoul, "Classification of Software and Hardware Bio-inspired Systems," ACS/IEEE AICCSA, pp. 1023-1028, 2006.

[5] S. F. Gilbert, Developmental biology, Seventh Edition, Sinauer associates Inc. Publishers, March, 2003.

[6] Junichi Suzuki, Tatsuya Suda, "A Middleware Platform for a Biologically Inspired Network Architecture Supporting Autonomous and Adaptive Applications," IEEE Journal on Selected Areas in Communications, Vol.23, No.2, February 2005, pp. 249-260, 2005.

[7] Sungyong Park, Bio Inspired System Software, Report of the New Technology Development for the Next Generation, Ministry of Commerce, Industry and Energy, Korea, 2005.

[8] R. I. Kondor and J. Lafferty, "Diffusion Kernels on Graphs and Other Discrete Input Spaces," Proc. of the International Conference on Machine Learning, 2002.

[9] C. Wei, J. Li, R. E. Bumgarner, "Sample size for detecting differentially expressed genes in microarray experiments," BMC Genomics 5: 87. PMID 15533245, 2004.

문 주 선

2006년 서강대학교 컴퓨터학과(학사). 2006년~현재 서강대학교 컴퓨터공학과 석사과정. 관심분야는 멀티미디어, 이미지 검색, MPEG, 생태계 모방 계산모델, 분산처리

낭 종 호

1986년 2월 서강대학교 전자계산학과 졸업(학사). 1988년 2월 한국과학기술원 전산과 졸업(석사). 1992년 2월 한국과학기술원 전산과 졸업(박사). 1992년 3월~1992년 8월 한국과학기술원 정보전자 연구소(연구원). 1992년 9월~1993년 8월 일본 Fujitsu 연구소(방문연구원). 1993년 9월~현재 서강대학교 컴퓨터학과 교수. 관심분야는 멀티미디어 시스템, 동영상 검색, 동영상 분석, 병렬/분산 처리, 인터넷 컴퓨팅