이동 에이전트를 이용한 병렬 인공신경망 시뮬레이터

조 용 만[†] · 강 태 원[††]

요 약

이 논문은 이동 에이전트 시스템에 기반을 둔 가상의 병렬분산 컴퓨팅 환경에서 병렬로 수행되는 다층 인공신경망 시뮬레이터를 구현하는 것을 목적으로 한다. 다층 신경망은 학습세션, 학습데이터, 계층, 노드, 가중치 수준에서 병렬화가 이루어진다. 이 논문에서는 네트워크의 통신량이 상대적으로 적은 학습세션 및 학습데이터 수준의 병렬화가 가능한 신경망 시뮬레이터를 개발하고 평가하였다. 평가결과, 학습세션 병렬화와 학습데이터 병렬화 성능분석에서 약 3.3배의 학습 수행 성능 향상을 확인할 수 있었다.

가상의 병렬 컴퓨터에서 신경망을 병렬로 구현하여 기존의 전용병렬컴퓨터에서 수행한 신경망의 병렬처리와 비슷한 성능을 발휘한다는 점에서 이 논문의 의의가 크다고 할 수 있다. 따라서 가상의 병렬 컴퓨터를 이용하여 신경망을 개발하는데 있어서, 비교적 시간이 많이 소요되는 학습시간을 줄임으로서 신경망 개발에 상당한 도움을 줄 수 있다고 본다.

키워드 : 인공신경망, 오류역전파, 병렬 신경망 시뮬레이터, 이동 에이전트, 병렬분산 컴퓨팅

The Parallel ANN(Artificial Neural Network) Simulator using Mobile Agent

Yongman Cho[†] · Taewon Kang[††]

ABSTRACT

The objective of this paper is to implement parallel multi-layer ANN(Artificial Neural Network) simulator based on the mobile agent system which is executed in parallel in the virtual parallel distributed computing environment. The Multi-Layer Neural Network is classified by training session, training data, layer, node, and weight in the parallelization-level. In this study, We have developed and evaluated the simulator with which it is feasible to parallel the ANN in the training session and training data parallelization because these have relatively few network traffic. In this results, we have verified that the performance of parallelization is high about 3.3 times in the training session and training data.

The great significance of this paper is that the performance of ANN's execution on virtual parallel computer is similar to that of ANN's execution on existing super-computer. Therefore, we think that the virtual parallel computer can be considerably helpful in developing the neural network because it decreases the training time which needs extra-time.

Key Words : Artificial Neural Networks, Error Back-Propagation, Parallel ANN Simulator, Mobile Agents, Parallel Distributed Computing

1. 서 론

인공신경망(이하 신경망)은 인간두뇌의 정보처리 기능을 모방한 계산모델로서 기호처리 인공지능기법으로는 다루기 어려운 문제들을 해결하는데 매우 유용하다. 그러나 학습해야할 입력 데이터가 매우 많거나, 매우 많은 뉴런을 포함하는 신경망은 학습에 수반되는 계산비용 문제로 구현하기가 어렵다[1]. 따라서 이런 계산문제를 해결하기 위해 신경망 모델을 병렬로 처리하려는 연구가 많이 있었고, 신경망은 그 자체에 병렬성이 내포되어 있어서 매우 자연스럽게 병렬로 구현할 수 있다. 신경망이 전 세계적으로 문자, 영상, 음성인식 등과 같은 다양한 분야에 응용하려는 많은 연구가 고가의 병렬컴퓨터에서 수행되어지는 것에 비해서, 국내의 경우 그런 고가의 병렬컴퓨터가 흔히 사용될 수 있는 환경이 아직 아니다. 따라서 국내에서는 신경망 응용시스템의 개발에 많은 시간이 소비된다. 이런 이유로 신경망 연구 및 응용에 대한 양적이며 질적인 진전이 어렵게 되었고, 이에 문제해결을 위해 추가적인 비용이 매우 적고 쉽게 시스템을 구현할 수 있는 이동에이전트 시스템을 이 논문에서 제안한다.

이 논문에서 제안한 시스템은 현재 우리나라의 IT환경과

※ 본 논문은 정보통신부의 정보통신기초기술연구지원사업(정보통신연구진흥원)으로 수행한 연구결과입니다.
† 준 회 원 : 해양센서네트워크시스템기술연구센터 연구원
†† 정 회 원 : 강릉대학교 컴퓨터공학과 교수
논문접수 : 2006년 2월 10일, 심사완료 : 2006년 10월 31일

무관하지 않다. 오늘날 컴퓨터 네트워크 기술의 발달과 보급은 자바 언어와 다양한 스크립트 언어를 사용하여 웹(WWW)상에서 플랫폼에 독립적으로 실행되는 다양한 응용 프로그램을 쉽게 개발하고 공유할 수 있게 하였다. 특히, 네트워크를 통하여 데이터가 고속으로 교환될 수 있기에, 저가의 하드웨어를 이용하여 초고속 병렬분산 처리를 실현할 수 있게 된 것이다[2,3,4]. 이러한 클러스터링의 실현은 병렬처리를 보편화한다는 점에서 의의가 매우 크고, 또한 병렬 프로그램의 계산 부분을 빠르고 안정적으로 처리한다. 하지만, 프로그램 실행 중 컴퓨터간의 통신이 많은 부분을 차지한다는 문제점이 지적되고 있으며[2], 클러스터링 자체가 구조적으로 고정적이며, 중앙 집중적이어서 프로그램이 시스템에 종속되는 정도가 심하다. 따라서 현재로서는 클러스터링이 범용 병렬컴퓨터를 저가로 실현하는 전용시스템의 성격이 강하다.

분산 인공지능이라는 분야에서 비롯된 에이전트 개념은 환경에서 얻은 지식과 자신의 내부 지식을 바탕으로 추론하고, 추론의 결과로 환경에 영향을 미치며 또한, 사용자를 포함한 다른 에이전트와 상호 작용하는 소프트웨어를 말한다[5]. 그 중에서 특히, 네트워크를 돌아다니며 수행되는 에이전트를 이동 에이전트(Mobile agent)라고 하는데, 이것은 네트워크의 부하를 줄이고, 비동기적이고 자율적으로 실행되며, 동적으로 적응할 수 있고, 플랫폼에 독립적이라는 장점을 갖는 것으로 대단히 많은 연구가 진행되고 있는 최신 기술이다[6,7,8]. 이러한 이동 에이전트에서 "이동"이라는 측면을 이용하여 네트워크상에 분산되어 존재하는 호스트들로 구성된 가상의 병렬분산 컴퓨팅 환경을 구축할 수 있으며, 이 환경은 "에이전트"가 갖는 자율성, 지능, 및 협동 능력으로 인하여 클러스터링과는 다르게 대단히 유연한 특징을 갖는다. 이 가상병렬 컴퓨터에서 신경망을 병렬로 처리하는 방법을 제시한다. 따라서 이동에이전트를 이용하면 하드웨어에 관계없이 에이전트의 최대장점인 융통성을 가지고 다양한 문제에 적용이 가능하다.

이 논문의 구성은 다음과 같다. 먼저 2장에서는 신경망을 병렬로 구현하는 기존의 접근방식과 이동에이전트에 관해서 살펴보고, 3장에서는 이 논문의 결과인 이동에이전트에 의한 학습세션 및 학습데이터를 병렬로 수행하는 인공신경망 시뮬레이터에 대해서 설명하고, 4장에서는 병렬 신경망 연구 분야의 대표적 벤치마크인 NetTalk을 사용하여, 개발한 시뮬레이터를 기존의 연구결과들과 평가·비교해보고 5장에서는 향후 연구 방향에 대해서 기술하고, 마지막으로 6장에서 결론을 맺는다.

2. 관련연구

여기서는 인공신경망을 병렬로 구현하는 접근방법과 신경망을 병렬로 구현하는 경우 병렬화의 대상에 따른 기존의 연구들을 살펴본다. 그리고 분산 인공지능이라는 분야에서 비롯된 에이전트 개념이 기존의 분산 시스템들과의 차이점과 이동에이전트의 특징을 설명한다.

2.1 신경망 병렬구현

이 절에서는 신경망 병렬구현의 배경과 신경망의 병렬화 대상에 따른 분류를 아래의 하위 절에서 다루도록 한다.

2.1.1 신경망 병렬구현의 배경

신경망에 대한 최고의 권위자인 Rumelhart와 McClelland가 신경망 모델을 PDP(Parallel Distributed Processing)모델로 불렀듯이, 원칙적으로 신경망이 병렬로 처리되어야 한다는 것은 이 분야의 연구자라면 모두 알고 있을 것이다. 신경망을 병렬로 구현하는 접근방법에는 범용의 병렬컴퓨터를 이용하는 방법과 특별한 전용 신경망 하드웨어를 사용하는 두 가지로 구분된다[1,9,10,11]. 특별한 전용 신경망 하드웨어는 특정 문제에 적합한 신경망을 하드웨어로 구현하는 것으로 가장 효과적이기는 하지만, 다른 문제에 사용할 수 없으며, 확장이 어렵고, 새로운 알고리즘이 개발되면 자체가 무용지물이 될 수 있다. 또한 신경망 응용 시스템을 개발하는 과정에 사용되기 어려운 접근방법이다. 따라서 이 논문은 신경망 응용 시스템을 개발하거나 신경망 학습 알고리즘을 개발하는 경우에 효과적으로 사용할 수 있는 소프트웨어적인 병렬 환경을 제공하려는 것을 연구하는 것이므로 하드웨어 기술에 관해서는 다루지 않는다. 범용 병렬컴퓨터를 사용하는 소프트웨어적인 방식은 병렬처리에 대한 중앙제어가 있느냐 없느냐에 따라 제어병렬(control parallel)과 데이터 병렬(data parallel)로 나뉜다.

데이터 병렬프로그램들은 분산되어 있는 데이터를 중앙집중식으로 처리한다는 특징을 갖는다. 모든 프로세서는 동일한 프로그램을 실행시키며 동기화는 중앙에서 통제된다. 이 방식은 크게 거친 구조(coarse structuring), 조밀 구조(fine structuring) 및 파이프라이닝(Pipelining) 세 가지로 구분된다. 다음에서 좀 더 자세히 살펴본다.

• 거친 구조(coarse structuring)

Zhang 등은 커넥션 머신(Connection Machine : CM)을 사용하여 오류 역전파 신경망의 계층별 뉴런 및 학습데이터 수준의 병렬 구현을 연구하였다[12]. 여기서는 각 계층의 뉴런들 일부가 하나의 프로세서에 의해서 처리되며 가중치들은 32개의 프로세서가 공유하는 공유메모리에 저장된다. 64K개의 프로세서를 갖는 CM에 대하여 NetTalk을 실험한 결과 연결 가중치를 초당 38M(million)회 수정할 수 있었다. 이와 유사한 연구사례로 MasPar 구현이 있다[13]. 여기서 각 프로세서들은 1차원 배열형태로 구성되어 신경망의 한 계층을 처리한다. 16K개의 프로세서를 갖는 MarPar에 대해서 NetTalk을 실험한 결과 연결 가중치를 초당 42M(million)회 수정할 수 있었다.

• 조밀 구조(fine structuring)

Rosenberg 등은 CM을 사용하여 오류 역전파 신경망의 뉴런 및 가중치 수준에서의 병렬 구현을 연구하였다[14]. 여기서는 3개의 프로세서가 한 조가 되어 한 뉴런의 입력, 처

리, 출력을 담당하도록 구성된다. 전진전파 단계에서 각 뉴런의 출력 담당 프로세서는 뉴런이 내보낸 출력에 자신의 가중치를 곱하여 자신에 연결된 뉴런들의 입력프로세서들에게 방송하고, 입력 프로세서들은 그 합을 계산한 후, 자신의 처리 프로세서에 그 값을 전달한다. 처리 프로세서는 활성함수(activation function)를 적용하여 자신의 출력을 계산하여 출력 프로세서에 전달한다. 역전파 과정 역시 유사하게 수행된다. 64K개의 프로세서를 갖는 CM에 대해여 NetTalk을 실험한 결과 연결 가중치를 초당 13M회 수정할 수 있었다.

• 파이프라이닝(Pipelining)

이 방식은 대개 고속의 계산을 처리하기 위하여 고안된 시스톨릭 어레이에서 구현된다. Pomerleau 등은 프로세서가 시스톨릭 링으로 구성된 Warp 시스템에 대하여 오류 역전파 알고리즘을 구현하였다[15]. 전진전파 과정에서 출력 값은 링을 따라서 원형으로 전달되며 대응되는 가중치와 곱해진 후, 가중치 합이 구해지면 활성함수가 적용된다. 역전파 과정 역시 유사하게 진행된다. 단지 10개의 프로세서를 갖는 Warp에 대하여 NetTalk을 실험한 결과 연결 가중치를 초당 17M회 수정할 수 있었다. 실제로 Chung 등은 13K(kilo)개의 프로세서로 구성된 시스톨릭 어레이를 사용하여 동일한 신경망에 대하여 연결 가중치를 초당 248M회 수정할 수 있다는 결과를 얻기도 하였다[16].

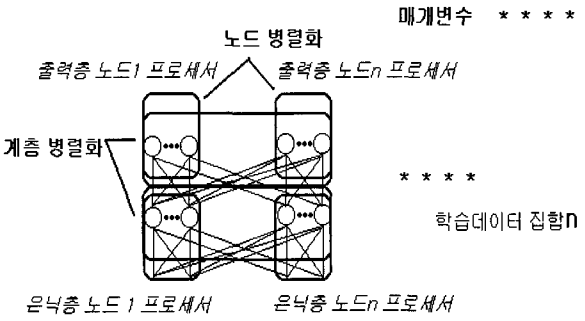
제어 병렬화는 분산된 데이터를 독자적으로 처리한다는 특징을 갖는다. 즉, 병렬 프로그램은 상이한 작업들로 명시적으로 분할되어 상이한 프로세서에서 실행되며, 각각의 프로그램은 일반적으로 서로 다른 것이며 통신은 일반적인 라우팅에 의존한다. 이 방식이 신경망에 적용된 사례로는 Straub 등의 연구가 있다[17]. 여기서는 신경망을 세로방향(각 계층을 수평으로 나열한 경우)으로 분할하는 방식을 사용하였다.

2.1.2 신경망의 병렬화 대상에 따른 분류

신경망을 병렬로 구현하는 경우 병렬화의 대상에 따라 학습세션(training session) 병렬화, 학습데이터(training example) 병렬화, 계층(layer) 병렬화, 노드(Neuron) 병렬화 및 가중치(weight) 병렬화 5가지로 나뉜다. 그림 1에서는 신경망 병렬화의 수준을 나타낸다.

• 학습세션 병렬화

학습세션 병렬화는 (그림 1)에 있는 매개변수들-중간층 노드개수, 학습반복횟수, 학습계수, 그리고 관성(momentum) 계수 등-의 다양한 값을 갖는 다수의 신경망 구조를 다수의 프로세서가 실행하도록 하는 것으로 이동 에이전트를 기반으로 하는 경우 가장 효과적으로 실행될 수 있다. 이 수준의 병렬화가 효과적으로 이루어지려면 단순히 학습계수 혹은 신경망의 구조를 달리하여 실행하는 것보다는 일정한 전략에 의하여 상이한 신경망을 만들거나, 적응적인 방식을 사용해야 한다. 이 부분을 이동 에이전트로 구현할 경우 에



(그림 1) 신경망 병렬화의 수준

이전트의 이동성(Mobility)을 발휘해서 인터넷에 연결된 컴퓨터들 중 이용가능한 자원이 많은 컴퓨터로 이동하여 신경망 학습을 수행한다. 또한, 수행완료 후 수행 결과를 제어에이전트로 보내야 할 때, 네트워크 에러가 발생할 경우 에이전트의 자치성(Autonomy)과 지능성(Intelligency)을 가지고, 네트워크가 회복될 때까지 일정한 전략에 의해 다른 신경망을 학습할 수 있다.

• 학습데이터 병렬화

학습데이터 병렬화는 상이한 학습데이터 집합을 상이한 프로세서들이 처리하도록 하는 것이다. 신경망 학습을 처리하는 각 호스트들은 자신이 학습하는 신경망의 가중치 수정량을 계산하여 반환하면 제어 에이전트는 이를 수집하여 가중치를 수정한 후, 각 신경망 에이전트들에게 수정된 가중치를 전송한다. 이때, 특정 호스트에서 수정 량을 상당기간 반환하지 않으면 그 호스트를 무시하고 해당 호스트의 학습데이터를 다른 호스트들에 반환하거나, 반대로 가용 호스트 개수가 증가하면 상황에 따라 학습데이터를 일정량 회수하여 추가된 호스트가 처리하도록 하는 등의 에이전트가 갖는 고유의 융통성을 활용할 수 있다.

• 계층 병렬화

계층 병렬화는 신경망의 계층들이 동시에 처리되는 것을 말한다. 예를 들어, 은닉층과 출력층이 상이한 프로세서에 의해서 처리되는 것을 말한다. 계층 병렬화는 파이프라이닝으로 구현할 것이며, 이 경우 출력층 프로세서가 출력과 신경망의 오차를 계산하는 동안 은닉층 프로세서는 다음 학습데이터를 처리하는 방식으로 수행되도록 한다. 계층 병렬화는 전 단계의 병렬화들과는 다르게 상대적으로 에이전트 사이의 통신이 증가된다. 따라서 각 호스트는 신경망 자체의 계산과 통신을 잘 조직하여야한다. 일반적으로 이 수준의 병렬화는 독자적으로 사용되는 경우보다는 학습데이터 병렬화 등과 결합되어 사용될 때 그 효과가 훨씬 크다.

• 노드 병렬화

노드 병렬화는 각 학습데이터에 대하여 노드들이 동시에

처리되는 것이다. 노드 병렬화의 개념을 사용하더라도 실제의 구현 사례들은 하나의 프로세서가 다수의 노드들을 포함한다. 즉, 다수의 노드들을 한 프로세스에 할당하는 방식의 노드 병렬화를 구현한다. 한 호스트에 할당되는 노드의 수는 대응되는 신경망의 구조 및 가용 호스트의 수에 의하여 결정한다. 또한, 에이전트 사이의 통신량이 다른 방식들보다 많으므로 효과적인 데이터 전달 방식뿐 아니라 한 호스트에 할당되는 노드의 수가 통신에 따른 오버헤드를 지나치게 크게 하지 않도록 조절해야한다.

이 논문에서는 이동 에이전트 사이의 통신량이 비교적 적은 학습세션 및 학습데이터를 병렬화 하여 처리하는 인공신경망 시뮬레이터를 개발하고 평가한다.

2.2 이동에이전트

이동 에이전트는 네트워크 에이전트 또는 순회 에이전트라고 부르며, 프로그램 자체가 네트워크를 돌아다니며 수행되는 것을 말한다[5,18]. 이동 에이전트는 분산 시스템을 개발하는데 있어서 기존의 방법과는 획기적으로 다르다. 분산 컴퓨팅을 위한 방법은 클라이언트-서버에서 출발하여 요구 코드(code-on-demand)를 거쳐, 현재는 이동 에이전트에 의한 개발 방법을 지향하는 것이 국제적인 추세이다[4,18,19].

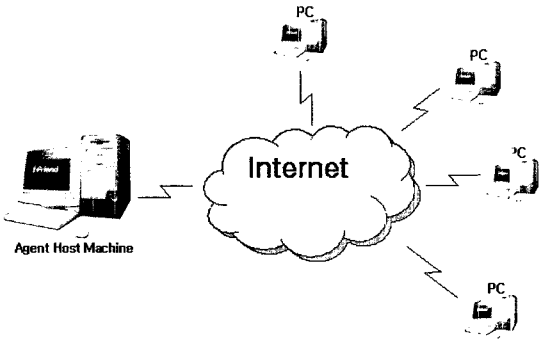
지금까지 이동 에이전트를 분산 시스템에 이용할 수 있는 많은 시스템들이 개발되었고 또한, 개발 중에 있다. 그 중에서 이 논문은 IBM의 이동 에이전트 시스템인 Aglets을 플랫폼으로 인공신경망 시뮬레이터를 개발하였다. 자바에 기초한 Aglets은 기존의 애플릿 프로그래머들이 쉽게 이동 에이전트 응용을 개발할 수 있도록 고안되었으며, 자바 가상 머신(JVM: Java Virtual Machine)이 설치된 호스트라면 기종이나 운영체제에 무관하게 사용할 수 있다. 그리고 IBM의 Aglets 팀이 개발한 에이전트 전송 프로토콜(ATP: Agent Transfer Protocol)은 분산 에이전트 시스템을 위한 응용 수준의 프로토콜로서 네트워크상의 호스트들로 이동 에이전트를 전송하는데 사용된다. 특히, 이것은 인터넷을 목표로 단순하면서 플랫폼에 독립적인 에이전트 이동을 제공한다. 따라서, 이 논문에서와 같이 기본적으로 인터넷에 기반을 둔 대규모 병렬분산 컴퓨팅 환경을 구축하는데 효과적일 것이며, 실제로 많은 이동 에이전트 개발에 사용되고 있는 시스템이다.

3. 이동 에이전트에 의한 병렬 인공신경망 시뮬레이터

개발 시스템은 학습세션 병렬과 학습데이터 병렬 시뮬레이터를 단일 인터페이스에서 사용하도록 구현하였다. 개발한 시뮬레이터의 하드웨어 및 소프트웨어 측면의 시스템 구성은 다음과 같다.

3.1 시스템의 하드웨어 구성

학습세션 병렬 인공신경망 시뮬레이터는 (그림 2)와 같이 물리적으로나 논리적으로 인터넷에 연결되어 있는 개개의



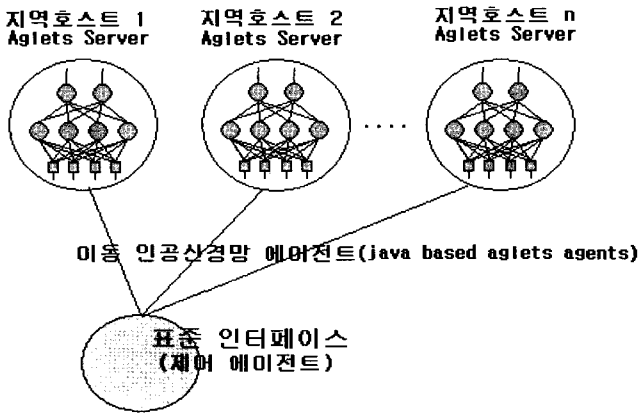
(그림 2) 시뮬레이터의 하드웨어 구성도

PC가 있고, 신경망 학습을 할 수 있는 이동에이전트들을 각 PC로 이동시켜서 각 PC의 자원들을 이용해서 신경망 학습을 한다. 그리고 결과를 다시 이동 에이전트 호스트(제어 에이전트가 실행되는 호스트)로 보낸다. 그리고 학습데이터 병렬 인공신경망 시뮬레이터는 각각의 이동 에이전트들이 독자적으로 학습한 가중치 수정량을 서로 주고받아야 하는데 필요한 네트워크 위상(topology)이 존재한다. 대표적인 네트워크 위상으로는 망(mesh) 구조, 링(ring) 구조, 버스(bus), 성(star)구조 등이 있다. 망(mesh)구조의 경우 제이 에이전트가 각 호스트(신경망)의 학습결과를 모두 넘겨받아 이를 각 호스트로 재분배하여 가중치를 수정하는 방법을 사용하는데, 이를 실제로 구현한 후 평가한 결과 이러한 구현은 통신량 뿐 아니라 대기시간이 많아서 비효율적임을 알 수 있었다. 이에 논문에서는 학습데이터 병렬화의 경우 네트워크 위상을 링(ring) 구조로 구현하였다. 즉, 각 호스트는 한 쪽에서 수정량을 전달받아 이를 자신의 수정량과 합산하여 가중치를 수정하고 동시에 그 값을 옆으로 전달하는 방식을 사용하였다. 가장 마지막 호스트는 실제로 모든 학습데이터를 사용한 가중치 수정량을 이용하여 자신의 가중치를 수정할 수 있다.

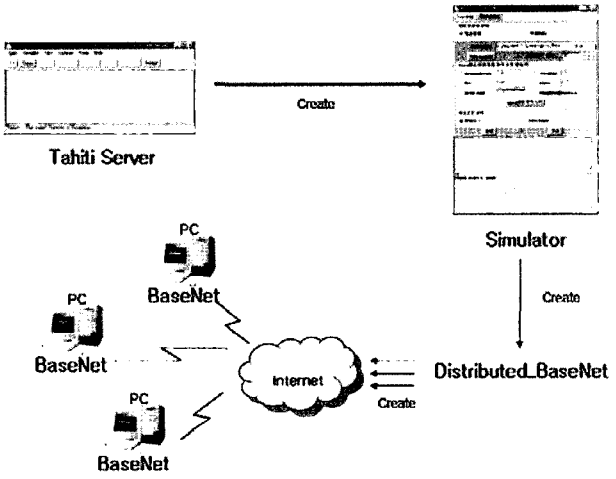
3.2 시스템의 소프트웨어 구성

학습세션 및 학습데이터 병렬 인공신경망 시뮬레이터는 에이전트 호스트에서 실행되고, 여기에서 여러 개의 이동 에이전트들을 에이전트 전송 프로토콜(ATP)을 이용해서 네트워크상의 분산된 컴퓨터로 전송한다.

개발 시스템의 사용 환경은 컴퓨터가 인터넷에 연결되어 있고 자바가 설치되어 있는 환경이면 개발한 병렬 인공신경망 시뮬레이터를 사용할 수 있고, 추가로 Aglets을 실행시킬 Aglet Server가 있어야 한다. 운영체제로는 Windows XP를 자바 개발 환경의 버전은 J2SDK1.4.1을 사용하였다. 구현을 위한 이동 에이전트 플랫폼은 Aglets을 사용하였으며 병렬 인공신경망 시뮬레이터의 구조는 다음 (그림 3)과 같다. 즉, 표준 인터페이스를 제공하는 제어 에이전트가 있고, 사용자는 이를 통하여 신경망의 구조 및 여러 학습 매개변수와 실행 호스트를 입력하여 지정한다.



(그림 3) 신경망 시뮬레이터의 구조



(그림 4) 에이전트 객체들의 생성관계

제어 에이전트는 입력받은 데이터를 기반으로 신경망과 이동 에이전트를 생성하여 정해진 호스트로 전송하면 각 호스트는 각자의 신경망을 학습한다. 학습이 끝나면 결과가 제어 에이전트로 전송되고 개발자는 그 결과를 분석하여 가장 효과적인 신경망 시스템을 만든다. 병렬 인공신경망 시뮬레이터는 15개의 자바 클래스들로 구성되어 있으며, (그림 4)에서는 에이전트 객체들의 생성관계를 나타내고 있다. (그림 4)에서처럼 Simulator객체(제어 에이전트)는 Distributed_BaseNet객체(이동 에이전트 및 신경망을 전송)를 생성하고 Distributed_BaseNet객체는 BaseNet 객체(이동 에이전트 및 인공 신경망)를 생성한다. 이때 BaseNet객체가 바로 이동에이전트이며, 이것이 인터넷에 연결되어 있는 지역호스트로 전송되어 신경망 학습을 하게 된다.

다음은 시뮬레이터를 구현한 핵심 클래스들의 역할과 중요 메소드들을 요약한다.

• BaseNet 클래스

Aglet 클래스를 상속받은 이동에이전트 객체를 생성하며,

```
public class BaseNet extends Aglet{
        신경망 구조에 관련된 매개변수 선언 및 초기화;
                ......
        public boolean handleMessage(Message msg) {
        /* 외부로부터 오는 메시지들에 따라 수행하는 일
           을 선택한다. */
        }//end of handleMessage

        public void doTaskInit(Object arg) {
            //이동에이전트로 보내질 신경망들의
              매개변수를 초기화;
                ......
        }//end of doTaskInit()

        public void Learning(double eta, double alpha) {
            //신경망에서 학습 부분
                ......
        }//end of Learning()

        public void doTask() {
        //신경망을 포함한 이동에이전트들을 다른
          호스트로 보내는 작업을 수행
        }//end of doTask()

        public String doRecognize(String patternfile){
        //학습을 끝낸 신경망을 이용해서 인식작업을
          수행한다.
        }//end doRecognize()
}//end of BaseNet class
```

(그림 5) BaseNet 클래스 구현

신경망 코드를 가지고 있어서 학습과 인식작업 모두를 수행한다. 신경망을 위한 각종 변수들을 포함하며, Distributed_BaseNet 객체로부터 다른 이동에이전트들의 Proxy와 학습을 위한 학습데이타와 매개변수 등을 전달받는다.

setWeightfile메소드는 저장된 Weight파일을 읽어 들인다. Learning메소드는 신경망을 학습하는 것이고, Learning2와 calcuDelta메소드는 학습데이타 병렬화에 사용되는 것입니다. Recognition메소드는 인식작업에 사용된다. (그림 5)에 이 클래스의 구현 내용을 나타낸다.

• AddressListWindow 클래스

이 클래스는 이동에이전트들이 전송될 주소를 "atp//<localhost IP>:<port #>"형태로 입력받는 창을 생성한다.

• Distributed_BaseNet 클래스

학습세션 병렬화 및 학습데이타 병렬화에 따라 다른 메소드를 사용하게 되며, AddressListWindow 객체에 의해 저장된 주소로 BaseNet객체를 전송하는 작업을 수행하게 된다. 그리고 각 이동에이전트들이 수행한 결과를 모두 취합해서 저장하는 작업도 수행한다.

sessionParallel메소드는 pattern file을 읽어서 신경망의 입력Data형태로 바꾸는 작업을 수행하는 루틴을 포함하고, 이동에이전트를 전송하는 루틴도 포함한다.

```
public class Distributed_BaseNet extend Aglet{
        변수들의 선언 및 초기화;
                .....
        // 이동에이전트들이 수행할 신경망들의 초기치
           값들을 읽어 들인다.
        public void initiation(){ ..... };
                .....
        // 이동에이전트들을 생성해서 목적지로 보내는
           역할을 한다.
        public void onCreation(){
                initiation();
                if(학습세션 병렬화 or 학습데이터병렬화)
                        sessionParallel();
                else    dataParallel();
        };
                .....
        // 신경망의 학습세션을 병렬로 처리한다.
        public void sessionParallel(){
        // 패턴파일을 읽어서 신경망의 입력 데이터
           형태로 바꾸는 루틴을 포함한다.
                .....
        };
        // 신경망의 학습데이타를 병렬로 처리한다.
        public void dataParallel(){.....};
        // 패턴파일을 읽어서 신경망의 입력 데이터
           형태로 바꾸는 루틴을 포함한다.
                .....
        };
                .....
        // 이동에이전트들과의 메시지를 서로 교환하기
           위한 역할을 수행한다.
        public boolean handleMessage(){.....};
}
```

(그림 6) Distributed_BaseNet 클래스 구현

dataParallel메소드는 sessionParallel메소드처럼 pattern file을 읽어서 신경망의 입력Data형태로 바꾸는 작업을 수행하고, Data를 보낼 로컬 호스트의 주소 수만큼 균등하게 쪼개는 작업을 포함한다. (그림 6)에 이 클래스의 구현 내용을 나타낸다.

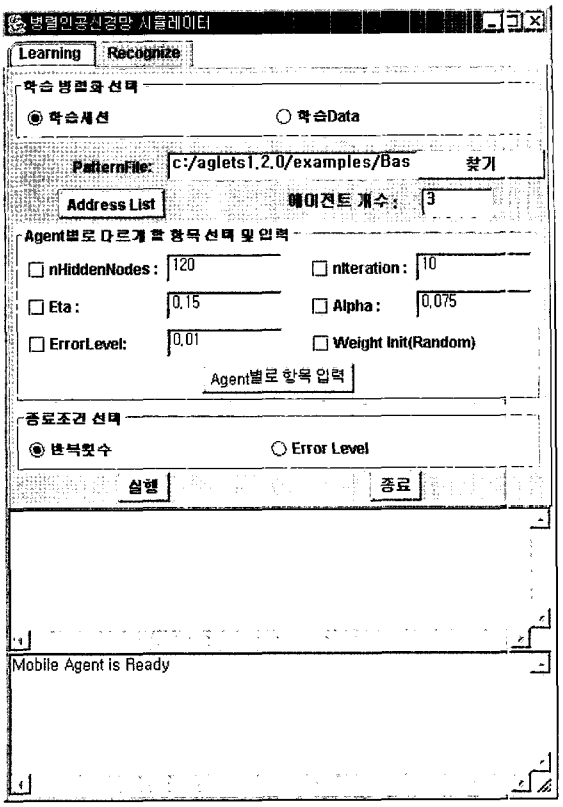
• Simulator 클래스

Tahiti Server(Aglet Server의 이름)에서 호출되는 클래스로 시뮬레이터를 생성한다. (그림 7)은 이 논문에서 개발한 시뮬레이터의 초기화면을 나타낸다. 이 시뮬레이터는 "학습세션 병렬처리", "학습데이터 병렬처리" 및 "신경망 학습 및 인식"을 수행할 수 있으며 인식 단계는 학습이 완료된 신경망을 테스트할 수 있는 것으로 현재는 인식 결과를 출력하는 수준이나 학습결과 및 인식결과를 그래프로 보여주도록 개선 할 수 있다.

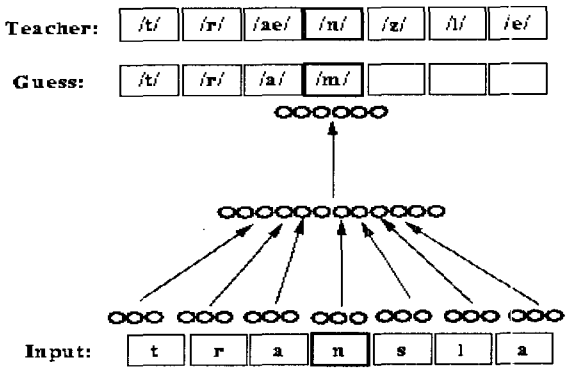
4. 실험에 의한 성능분석

4.1 성능분석 요소

평가를 위해서는 병렬 신경망 연구 분야의 대표적 벤치마



(그림 7) 병렬 인공 신경망 시뮬레이터의 초기화면



(그림 8) NetTalk의 구조(그림에서 입력뉴런은 각 문자마다 26개(총 182개)이며 출력뉴런은 52개)

크인 NetTalk을 사용하였다. 평가에 사용한 신경망은 입력뉴런 182(구두점 제외), 출력 뉴런 52(발음요소), 중간뉴런 10~120개를 사용하였다. NetTalk은 한 단어를 구성하는 각 문자의 발음을 찾아내는 것이다. 이를 위하여 NetTalk 학습 시에는 각 단어에 대하여 학습하려는 문자의 앞뒤 3문자를 포함 총 7문자가 입력으로 사용된다.

NetTalk의 구조를 (그림 8)에서 나타내고 있다. 평가를 위한 측도는 CUPS(connection update per second)를 사용하였다. 즉 1초당 신경망이 학습을 위하여 가중치 수정량을 몇 번 계산하느냐를 기준으로 하였다. 결과는 다음 <표 1>

〈표 1〉 가상의 병렬 컴퓨터의 성능 평가 결과

computer	호스트 수	신경망 크기	학습방법	MCUPS
P-IV 1.8GHz	4	NetTalk 182×120×52	lbe	5.9~6.5
〃	8	NetTalk 182×120×52	lbe	7.9~11.1
〃	3	NetTalk 182×10~120×52	lbp	9.8~27.0

lbe : learning by epoch
lbp : learning by pattern
MCUPS : Mega CUPS

과 같다(학습데이터 756개 기준, 네트워크 시간포함).

실험의 비교를 위해서 사용한 과거의 연구결과는 <표 2>에 나타낸 바와 같다[18]. 이들 연구와 비교함에 있어서 다수의 프로세서를 사용하는 병렬 컴퓨터와 직접적인 비교는 큰 의미는 없다고 판단된다. 그러한 컴퓨터들은 상대적으로 CPU 속도가 Pentium 보다 현저히 느린 대신 통신 오버헤드는 아주 작을 것이다. 그러나 실험결과에서 수백 개의 프로세서를 사용하는 정도의 성능을 얻을 수 있다는 것은 개발한 시뮬레이터가 효과적임을 알 수 있다. 또한 Sun SparcStation 10 (1.1MCUPS)에서의 결과와도 잘 들어맞는다. 실험에 사용한 P-IV는 Sun보다 약 9배 빠르며, 실험에서와 같이 3대에서 병렬로 실행한 경우 27.0MCUPS가 나온 것은 개발 시뮬레이터가 잘 수행되는 것을 반증한다.

다만, <표 1>에서와 같이 개발한 시뮬레이터의 경우 어떤 경우에는 3대에서 병렬로 수행하여도 9.8MCUPS가 나오는 경우가 있는데, 그 경우는 실험에서 중간노드의 개수가 적고 반복 횟수가 적어서 상대적으로 네트워크 오버헤드가 크기 때문에 발생된 수치로 파악되었다. 실제로 신경망의 규모가 크고 학습 횟수가 많을수록 성능이 안정적임을 확인할 수 있었다.

4.2 학습데이터의 크기에 따른 학습수행시간 성능 평가

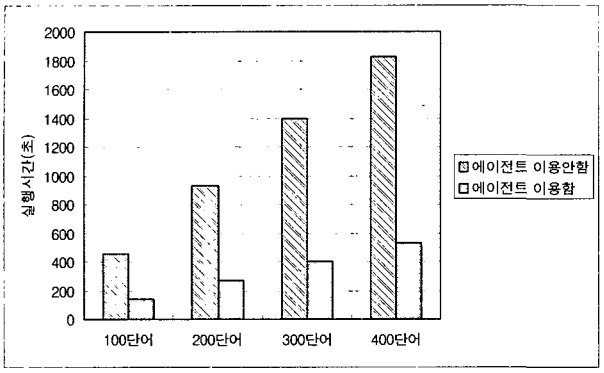
이 실험에서는 학습데이터의 크기를 달리해가면서 각각의 수행속도를 측정하였다. 학습에 사용된 단어의 개수는 각각 100/200/300/400 단어, 중간 노드는 120개로 하였으며, 각각 10회씩의 수행을 하였고, 4개의 이동에이전트를 이용해서 4대의 컴퓨터에서 수행하였다. 신경망을 이용한 NetTalk 학습은 앞에서 설명한 것과 같이 단어의 개수가 학습데이터의 개수는 아니며 실제로는 단어에 포함된 문자의 개수가 학습데이터의 개수이다. 실제로 실험에 사용한 단어 100/200/300/400을 위한 학습데이터의 개수는 756/1483/2219/2903으로 단어수의 약 7배이다. 하지만 편의를 위하여 실험의 분석에서는 학

〈표 2〉 과거의 연구로 알려진 평가 결과

Computer	Number of PEs	Network Size	μ	FP	MCUPS
Sun-3[19]	1	Nettalk	1	32	0.034
NCube/4+[5]	16	Optimal network	1	32	0.19
Sun SparcStation 10[84]	1		1	32	1.1
Alpha Station[84]	1		1	32	3.2
CM-2[85]	16K	Nettalk(60)	1		2.8
Cray2[22]	4	257×256×131,072	1	32	101
iPSC/860[19]	32	Nettalk(80)	2,000	32	11
MP-1[14]	4,096	128×64×16	1,536	32	12
IMB RISC/6000 550 [86]	1	1000×1000×1	1,000	32	17.6
Cray X-MP [22]	4	257×256×131,072	1	32	181
CM-5[23]	32	Nettalk(80)	lbb	32	18.33
CM-2[21]	65,536	Nettalk(80)	4,096	32	40
Cray Y-MP[22]	2	257×256×131,072	1	32	401
MP-1216 [15]	16,384	Nettalk(120)	lbb	32	41
Q-x system [28]	64	257×128×256	8,196	32	441
CM-5 [11]	256	1024×256×64	lbe	32	50
Fujitsu VP-2400/10 [25]	1	Nettalk(60)	1	32	60
CM-5[22]	512	257×256×131,072	1	32	76
AP1000 [19(in Chapter 7)]	512	Nettalk(120)	906	32	86
CM-2 [87]	64K	128×128×128	65,536	32	350
TLA[10]	16	256×64×256	1	32	0.6
Transputers[54]	53	Nettalk(60)	1,500	32	1.54
MEIKO [51]	120	512×256×64	30	32	8
Sandy [62]	32	Nettalk(60)	1	32	42
Cellular arch. [88]	4,175	Nettalk(60)	1	16	51.51
SNAP [79]	64	64×64×64		32	76.6
RAP [59]	40	640×640×640	1	32	102
MUSIC [6]	45		1	32	203
SNAP[79]	64	5124×512×512		32	302
Sandy [62]	256	Optimal network	1	32	567
MY-NEUPOWER [74]	512	Optimal network		16	1,260
CNAPS [72,73]	512	1,900×500×12	1	16	2,3791

[1]The performance has been estimated or simulated (not measured).

통신오버헤드 = 도착시간 - 학습시간



(그림 9) 학습데이터의 크기를 달리해서 실험한 결과

〈표 3〉 통신 오버헤드 및 개발시간 비율 평가(단위 초)

학습 데이터 수(단어)	평균 실행시간	각 호스트들의 실행시간 합	시뮬레이터 총 수행시간	통신 오버헤드 비율(%)[1]	개발시간 비율(%)[2]
100	114.8	459.4	138.5	17.1	30.1
200	233.0	932.2	272.5	14.5	29.2
300	348.6	1394.5	402.0	13.3	28.8
400	456.4	1825.6	531.3	14.1	29.1

습데이터의 개수는 단어수로 표현하기로 한다.

(그림 9)는 이 실험의 결과를 나타내는 것으로서 '실행시간'에서 병렬 인공신경만 시뮬레이터를 이용해서, 이동에이전트가 다수의 컴퓨터(4대)에서 같은 양의 데이터를 동시에 처리하므로 수행시간이 이론적으로는 4배가 되어야 하지만, 통신오버해드와 문제 분할등과 같은 전처리 오버헤드로 인해 약 3.3배의 정도 빠른 수행속도를 보인다. <표 3>은 이 결과를 수치상으로 보여주는데, 병렬화에 따른 통신 오버헤드와 본 연구의 결과물인 시뮬레이터를 사용하여 신경망을 학습하는 경우의 신경망 학습시간 단축비율을 나타낸다. 표에서 알 수 있듯이 통신 오버헤드는 대략 15% 정도이며 이 비율은 신경망의 규모가 커질수록 감소할 것이다. 신경망 학습에 소비되는 시간은 에이전트를 이용하지 않는 경우에 비해 평균 30% 정도의 시간으로 처리가 됨을 보여준다. 즉 이동에이전트를 이용하면 3.3배 이상의 수행속도 향상을 가져오는데 이는 더 많은 컴퓨터에서 처리할수록 더 높은 수행속도 향상을 가져옴을 알 수 있다.

통신오버헤드는 전용병렬컴퓨터에 비해서 이동에이전트를 이용하게 되면 이동에이전트가 신경망 학습을 위해서 신경망을 실행하기 전에 원거리에 있는 컴퓨터로 전송되는 시간과 모든 신경망 학습을 종료하고 그 결과를 다시 보내오는데 소비되는 시간을 말한다. 따라서 다음 식으로 표현될 수 있다.

4.3 학습세션 병렬화 성능분석

신경망 학습을 위해서는 여러 가지 매개변수를 정해주어야 하며 이러한 매개변수 값들은 신경망의 학습에 영향을 준다. 하지만 특정 문제에 가장 적절한 매개변수를 정하는 규칙은 아직까지 알려진 것이 없다. 따라서 신경망의 학습과 성능을 최적으로 하기 위해서는 여러 번의 실험을 통해서 휴리스틱하게 최적의 매개변수 값을 찾을 수밖에 없으며, 여기에는 많은 시간과 노력이 필요하게 된다. 이러한 단점을 극복하기 위한 하나의 선택이 바로 병렬 인공신경망 시뮬레이터를 이용해서, 다양한 매개변수로 신경망을 동시에 학습하여, 신경망 응용 시스템 개발을 위한 시간과 노력을 최소화하는 것이다.

시뮬레이터에서 각 이동에이전트별로 입력 가능한 매개변수는 중간층 노드개수, 학습반복횟수, 학습계수(η), 그리고 관성(momentum)계수(α) 등이다. 첫 번째 학습세션 병렬화 실험으로서, 매개변수 중에 중간층노드의 개수를 달리했을 때의 실험결과를 <표 4>에서 보여주고 있다. 이 실험에서는 시뮬레이터에 의해서 8대의 컴퓨터에서 동시에, 다른 중간 노드개수를 가진 신경망들의 실행결과를 보여준다. 이 실험에서는 중간층 개수가 40일 때 가장 좋은 오류수준(ErrorLevel)을 나타내고 있다.

두 번째 실험에서는 첫 번째 실험에서 중간층 노드의 개수를 40으로 했을 때, 가장 좋은 결과를 나타냈으므로 중간층의 개수를 40으로 하고, 학습계수를 변경해서 8대의 컴퓨터에서 이동에이전트를 실행 시켰다. <표 5>에는 두 번째 실험결과를 나타낸다. 이 실험에서는 학습계수(η)가 0.1의 값을 가질 때 가장 좋은 결과를 보였다.

세 번째 실험에서는 중간층 노드의 개수가 40이고, 학습계수(η)는 0.1일 때 관성계수(α)를 달리해서 실험한 결과를 <표 6>에서 나타냈다. 이 실험에서는 관성계수가 0.075일 때 가장 좋은 결과 값을 나타냈다.

위에서 실시한 3가지 실험을 하는데 걸린 총 실행시간은 322.5초다. 이 시간에는 이동 에이전트를 로컬 호스트로 전

〈표 4〉 중간층 노드의 개수를 달리해서 학습세션 병렬화를 실행한 결과

에이전트#	중간층 노드개수(#)	반복횟수	η	α	ErrorLevel	실행시간 (밀리초)
1	20	10	0.15	0.075	0.019215	13203
2	30	10	0.15	0.075	0.01923	22250
3	**40**	10	0.15	0.075	**0.01917**	30578
4	50	10	0.15	0.075	0.019181	39203
5	60	10	0.15	0.075	0.019199	49625
6	80	10	0.15	0.075	0.019229	68485
7	100	10	0.15	0.075	0.019231	84985
8	110	10	0.15	0.075	0.019231	95921

〈표 5〉 학습계수(η)를 달리해서 학습세션 병렬화를 실행한 결과

에이전트#	중간층 노드개수(#)	반복횟수	η	α	ErrorLevel	실행시간 (밀리초)
1	40	10	**0.1**	0.075	**0.01911**	31828
2	40	10	**0.2**	0.075	0.019203	31391
3	40	10	**0.3**	0.075	0.019225	31500
4	40	10	**0.4**	0.075	0.019228	31578
5	40	10	**0.5**	0.075	0.019229	31515
6	40	10	**0.6**	0.075	0.019229	31625
7	40	10	**0.7**	0.075	0.019226	31391
8	40	10	**0.8**	0.075	0.019229	30578

1) 통신오버헤드 비율(%) = 통신오버헤드 시간 / 총 수행시간

2) 개발시간비율(%) = 이동에이전트를 이용하여 가상의 병렬 컴퓨터에서 신경망을 수행한 시간 / 이동에이전트를 이용하지 않고 한 컴퓨터에서 신경망을 수행한 시간

〈표 6〉 관성계수(α)를 달리해서 학습세션 병렬화를 실행한 결과

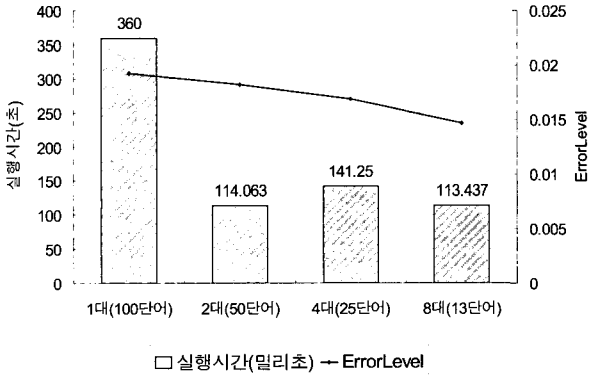
에이전트#	중간층 노드개수(#)	반복횟수	η	α	ErrorLevel	실행시간 (밀리초)
1	40	10	0.1	0.01	0.019117	31844
2	40	10	0.1	0.02	0.019116	31734
3	40	10	0.1	0.03	0.019114	30484
4	40	10	0.1	0.04	0.01911	31344
5	40	10	0.1	0.05	0.01911	31625
6	40	10	0.1	0.06	0.019108	31828
7	40	10	0.1	0.07	0.019108	31422
8	40	10	0.1	0.08	0.019108	31969

송하고 결과 값을 받는데 걸리는 네트워크 오버헤드가 포함되었다. 그러나 한 대의 컴퓨터에서 이와 같은 실험을 실시했을 때 걸리는 총 실행시간은 907.9초이므로 병렬 인공신경망 시뮬레이터를 이용했을 때 약 2.8배의 효율성이 있음을 알 수 있다.

4.4 학습데이터 병렬화 성능분석

이 평가에서는 학습데이터 100단어(학습데이터 756개)를 각각 2, 4, 8대의 호스트에서 병렬로 처리하는 학습데이터 병렬화의 성능을 평가한다. 예를 들어 2대의 호스트에서 수행하는 경우 각각의 호스트는 50개씩의 단어를 가지고 학습을 하며, 각각의 호스트들은 Epoch단위로 가중치 수정 량을 계산하여 다른 호스트에 이를 전송한다. 실험에서 중간 노드의 개수는 40으로 하였으며 학습반복 횟수는 100번 수행하는 것으로 하였다. 또한 데이터교환을 위한 논리적 연결이 망 구조일 경우, 가중치 수정량을 교환하는데 소비되는 시간이 지나치게 과다한 것으로 판단되어 이 시뮬레이터는 학습데이터 병렬화에 링 구조를 사용하였다.

실험의 결과는 (그림 10)과 같다. 결과에서 알 수 있듯이 링 구조에서의 병렬화는 명백한 이득을 얻을 수 있었다. 한 대의 컴퓨터에서 학습하는 경우 학습시간은 약 362.4초이며, 2/4/8 대의 호스트를 사용하여 학습데이터를 병렬화 하여 학습하는 경우 각각 약 113.5/114.3/111.6 초의 학습시간이 소요된다. 따라서 각각 319%, 317%, 325%의 학습수행시간 효율을 확인할 수 있었다. 즉, 3.19배/3.17배/3.24배 수행성능



(그림 12) 링 네트워크위상에서 학습데이타 병렬화를 수행한 결과 차트

향상을 가진다. 이러한 결과는 단순히 학습수행시간만을 고려한 것이며, 각각의 실행에서 얻은 신경망의 오류수준이 각각 0.019, 0.018, 0.016, 0.014인 점을 고려한다면 병렬화가 한층 효과적일 수 있음을 시사한다.

5. 향후 연구 방향

향후 연구의 방향은 크게 두 가지로 요약된다. 첫째, 현재의 연구 결과는 특별히 다른 작업을 수행중이지 않은 단일 네트워크상의 동질적인 네트워크 호스트를 사용했으며, 이질적인 호스트들로 구성된 인터넷을 기반으로 손쉽게 그러나 시간에 구애받지 않고 병렬 처리를 하기 위하여, 이동 신경망 에이전트들이 네트워크의 상태와 자신이 수행되는 지역 호스트의 상태를 파악하고 대응하는 수단을 갖도록 해야 한다는 것이며 둘째, 불필요한 기능을 많이 가지고 있는 범용의 이동 에이전트 플랫폼이 아니라 인터넷 환경에서 병렬분산 처리를 손쉽게 하기 위하여 별도의 병렬분산 처리를 위한 이동 에이전트 플랫폼을 연구할 필요가 있다는 것이다. 이러한 연구는 기존의 그리드 컴퓨팅[21]분야와 매우 유사한 면이 있지만, 그리드 컴퓨팅은 슈퍼컴퓨터간의 네트워크를 통한 가상의 병렬컴퓨터 구현이라면, 이 논문에서는 개인 PC를 이용한 것이 차이라 하겠다.

6. 결 론

인공신경망은 응용 범위가 대단히 넓은 첨단 컴퓨터 기술이며, 생물학에서 그 동기를 가지고 온 것이다. 생물학적 이론에 바탕을 둔 많은 첨단 과학 기술이 속속 등장하고 그것의 유용성이 입증되는 시점에서 인공신경망을 응용하는 연구는 더욱 증대될 것이다. 또한, 저가의 고성능 컴퓨터의 보급이 확대되고, 가정에서도 초고속망을 사용하게 됨으로써, 상당한 컴퓨팅 자원이 낭비되는 것도 사실이다. 이러한 추세에서 그러한 컴퓨팅 자원을 고급 기술 연구 및 응용에 활용하는 것은 매우 중요하고 시급하다.

이 논문은 이동 에이전트 시스템에 기반을 둔 가상의 병렬분산 컴퓨팅 환경에서 병렬로 수행되는 다층 인공신경망 시뮬레이터를 구현하는 것을 목적으로 한다. 이 논문에서는 네트워크의 통신량이 상대적으로 적은 학습세션 및 학습데이터 수준의 병렬화가 가능한 신경망 시뮬레이터를 개발하고, 평가하였다. 개발한 병렬 시뮬레이터는 학습세션 병렬화의 경우 일체의 학습 매개변수를 사용자가 지정할 수 있도록 하였으며, 학습데이터 병렬화의 경우 각 호스트는 분배받은 학습데이터로 학습하되 다른 호스트들의 학습량을 전송 받아 자신의 학습에 반영한다.

NetTalk 문제에 대한 실험 결과, 개발한 시뮬레이터는 8대의 호스트(P-IV 1.8M CPU)에 대하여 lbe(learning by epoch) 학습의 경우 약 8~11MCUPS라는 우수한 속도로 수행됨을 확인 할 수 있었으며, 평균적으로 신경망을 개발하는데 드는 시간을 1/3로 줄일 수 있었다. 학습데이터의 크기에 따른 수행시간 성능 평가에서 통신 오버헤드는 약 15%

정도이며, 신경망 학습에 소비되는 시간은 평균 70%정도 감소되는 것으로 나타났다. 그리고 학습세션 병렬화 성능분석에서도 병렬 인공신경망 시뮬레이터를 이용했을 때 약 2.8배의 학습수행 성능 향상을 확인 할 수 있었고, 학습데이터 병렬화 성능분석에서도 약 3.3배의 수행속도 향상이 있음을 알 수 있었다. 이러한 결과는 단순히 학습수행시간만을 고려한 것이며, 각각의 실행에서 얻은 신경망의 오류수준을 고려한다면 병렬화가 한층 효과적일 수 있음을 알 수 있다.

이 논문의 결과는 고가의 병렬컴퓨터가 아닌 인터넷 환경에 기반을 둔 가상의 병렬컴퓨터에서 이동에이전트를 이용하여 수행하였으며, 이 가상의 병렬 컴퓨터에서 신경망을 병렬로 구현하여 기존의 전용병렬컴퓨터에서 수행한 신경망의 병렬처리와 비슷한 성능을 발휘한다는 점에서 이 논문의 의의가 크다고 할 수 있다. 따라서 가상의 병렬 컴퓨터를 이용하여 신경망을 개발하는데 있어서, 비교적 시간이 많이 소요되는 학습시간을 줄이므로 신경망 개발에 상당한 도움을 줄 수 있다고 본다. 국내의 연구 동향이 미진하기는 하나, 해외에서도 기반기술에 대한 연구 결과가 이제 막 실현되는 초기 시점에 있으며, 인터넷 인프라가 세계적 수준인 국내 환경을 고려한다면 이 논문의 결과는 신경망 분야 및 인터넷 환경에서의 병렬·분산 컴퓨팅의 활성화에 크게 기여할 것이다.

참 고 문 헌

[1] Nikola B. Serbedzija, "Simulating Artificial Neural Networks on Parallel Architectures", Computer, Vol. 29, No. 3, 56-63, 1996.

[2] Pacheco P., Parallel Programming with MPI, Morgan Kaufmann, 1997.

[3] William Gropp, Ewing Lusk, Anthony Skjellum, Using MPI: Portable Parallel Programming with the Message-Passing Interface, MIT Press, 1996.

[4] Danny B.L., Mitsuru O., Programming and Deploying Java Mobile Agents with Agelets, Addison-Wesley, 1998.

[5] Mario Tokoro, "Agents: Towards a Society in Which Humans and Computers Cohabitate", in Distributed Software Agents and Applications, Springer, pp. 1-10., 1995.

[6] T. R. Ioerger, R. A. Volz, J. Yen, "Modeling Cooperative, Reactive Behaviors on the Battlefield Using Intelligent Agents.", Proceedings of the Ninth Conference on Computer Generated Forces (9th CGF) , pp. 13-23, 2000.

[7] M. Miller, J. Yin, R.A. Volz, T.R. Ioerger, J. Yen, "Training Teams with Collaborative Agents.", Proceedings of the Fifth International Conference on Intelligent Tutoring Systems, (ITS-2000) , pp. 63-72, 2000.

[8] O. Badawy,; A. Almotwaly, "Combining neural network knowledge in a mobile collaborating multi-agent system" Electrical, Electronic and Computer Engineering, 2004. ICEEC apos; 04, 2004.

[9] Manavendra Misra, "Parallel Environments for Implementing Neural Networks", Neural Computing Surveys Vol. 1, 48-60, 1997. http://www.icsi.berkeley.edu/~jagota/NCS

[10] Gerd Kock and Nikola B. Serbedzija, "Simulation of Artificial Neural Networks", SAMS, Vol. 27, 15-59, 1996.

[11] Erich Schikuta, "Structural Data Parallel Neural Network Simulation", 1997, http://citeseer.nj.nec.com/29242.html.

[12] X. Zhang, M. Mckenna, J. P. Mesrov and D. L. Waltz, "The backpropagation algorithm on grid and hypercube architectures", Parallel Computing, Vol. 14, No. 3, 317-327, 1990.

[13] A. Zell, N. Mache, T. Sommer and T. Korb, "Recent Developments of the SNNS Neural Network Simulator", Proc. Applications of Neural Networks Conference SPIE, 708-719, 1991.

[14] C. R. Rosenberg and G. Blelloch, "An Implementation of Network Learning on the CM", Proc. Joint Conference Artificial Intelligence, 329-340, 1987.

[15] D. A. Pomerleau, G. L. Gusciora, D. S. Touretzky and H. T. Kung, "Neural Network Simulations at Warp Speed: How We Got 17 Million Connections per Second", Proc. IEEE ICNN, San Diego, 143-150, 1988.

[16] J-H. Chung, H. Yoon and S. R. Maeng, " A Systolic array Exploiting the Inherent Parallelism of Artificial Neural Networks", Microprocessing and Microprogramming, Vol. 33, No. 3, 145-159, 1991/92.

[17] R. Straub, D. Schwarz and E. Schoneburg, "Simulation of backpropagation networks on transputers", Neurocomputing, Vol. 2, No. 5&6, 199-208, 1991.

[18] D. Chess, C. Harrison, A. Kershenbaum, "Mobile Agents: Are they a good idea?", IBM T,J Watson Research Center, 1995.

[19] David Kotz, Robert S. Gray, "Mobile Agents and the Future of the Internet", http://www.cs.dartmouth.edu, 1999.

[20] N. Sundararajan, "Parallel Architectures for Artificial Neural Networks: Paradigms and Implementations", wiley pp. 49. 2002.

[21] Oh-Kyoung Kwon , Jaegyoon Hahm, Sangwan Kim, and Jongsuk Lee , "A Grid Resource Allocation System for Scientific Application: Grid Resource Allocation Services Package (GRASP)" Mardi Gras Conference, 2005.

조 용 만

e-mail : ymcho@kangnung.ac.kr

1996년 강릉대학교 전자계산학과(이학사)

2002년 강릉대학교 컴퓨터공학과(공학석사)

2005년 강릉대학교 컴퓨터공학과 박사과정수료

1996년~2001년 한전KDN(강릉지사) 근무

2002년~2004년 방송통신대학교 조교

2005년~현재 해양센서네트워크시스템기술연구센터 연구원

관심분야 : 인공생명, 인공지능, 복잡계, 센서네트워크

강 태 원

e-mail : twkang@kangnung.ac.kr

1985년 연세대학교 수학과(이학사)

1988년 고려대학교 전산학과(이학사)

1991년 고려대학교 수학과(이학석사)

1996년 고려대학교 컴퓨터학과(이학박사)

1996년~현재 강릉대학교 컴퓨터공학과 교수

관심분야 : 인공생명, 인공지능, 소프트 컴퓨팅, 복잡계