.NET을 기반으로 한 효율적인 전자상거래 시스템에 관한 연구

A Study on an Efficient e-business System based on .NET

나윤지
충북대학교 컴퓨터공학과 박사과정

Yun-Ji Na
Ph. D. Course, ChungBuk National University,
Dept. of Computer Engineering

중심어 : .NET, Component, Web Customer, Web Server, Response time, reference characteristics

요 약

전자상거래 사용자의 폭발적인 증가는 전자상거래 시스템의 급격한 로드 증가와 네트워크 트래픽을 증가시키고 있다. 또한 전자상거래 시스템에서 동영상을 포함한 자료가 점차 늘어나고 있고 이에 따라 인터넷 서비스 제공업자들은 시스템의 관리와 회선의 유지를 위해 막대한 비용을 지불하고 있다.

이러한 시스템의 로드와 네트워크의 트래픽 증가는 웹 고객의 서비스 요청에 대한 응답 속도의 지연을 가져오게 되며 이는 고객만족도를 감소시켜 e-비지니스 기업의 경쟁력을 떨어뜨리는 중대한 요인으로 작용하고 있다.

본 연구에서는 마이크로소프트사의 최신 버전인 닷넷을 기반으로 한 전자상거래 시스템에서, 응답 지연 속도를 개선한 전자상거래 시스템을 설계하고 실험을 통해 성능을 분석하였다. 실험 결과 제안한 시스템은 웹 고객의 다양한 요구에 대해 개선된 응답 속도와 빠른 적응성을 가질 수 있으며, 특히 전자상거래 시스템에 대해 동시에 발생하는 다수의 다중 사용자의 요구로 인해 발생되는 응답 속도의 지연을 막을 수 있을 것으로 기대된다.

Abstract

A electronic business user is increased rapidly, and load of a electronic business system and network traffic are increased rapidly. Also, the data which included an audio and video is increasing in a electronic business system gradually, and According to this an Internet service businessman is paying for an enormous cost for management of a system and a maintenance of a line. It is brought a delay of a response speed about a service wish of a Web customer this system load and network traffic raise. Also, this decreases customer satisfaction and becomes an important factor to decrease competitive power of an e-business enterprise.

In this study, we proposed the electronic business system that improved answer delay time about a Web customer in an e-business system based on the .net which was a the latest version of Microsoft company, and analyzed performance of a proposal system through an experiment. In the experiment result, the system that proposed it can have response time to have been improved and fast adaptability about various requests of a Web customer. Also, in a proposed system, a decrease gets a delay of a response speed to occur because of the multiple user done.

I. 서론

기존의 상거래가 표준화된 고객집단으로부터 단일한 요구를 얻어 단일품종으로 생산해 판매하던 대량생산-시장점유의 패러다임이었다면 전자상거래에서 구현되는 패러다임은 개별화된 일대일 관계를 유지하는 개별화-고객점유의 패러다임이라고 할 수 있다. 따라서 전자상거래의 발전은 기존에 마케팅 채널인 인간 대 인간, 전화, DM(Direct Mail)에 이어 새로운 채널로서 인터넷 마켓팅을 제공하게 되었다. 효율적인 전자상거래의 활용을 위해서는 사용자 중심의 인

터페이스를 제공해야하며 쉽고 빠르면서도 사용자가 만족할만한 수준의 상품정보를 제공할 수 있는 적응성을 가지고 있어야하며 사용자 측면이 고려된 시스템의 개발이 필수적이다. 이를 위해 지능적인 에이전트 기술을 도입한 전자상거래 지원시스템[1],[2]에 대한 연구와 사용자의 요구에 대한 응답 속도의 개선을 위한 각종 연구가 필요하다.

인터넷을 통한 전자상거래 사용자의 폭발적인 증가는 전자상거래를 위한 시스템의 급격한 로드 증가와 네트워크 트래픽을 증가시키고 있으며 동일한 객체에 대한 중복된 요청이 네트워크 대역폭의 상당 부분을 차지하여 불필요한 시스템 자원의 낭비를 초래하고 있다. 이에 따라 인터넷 서비스 제공업자(ISP:Internet Service Provider)들은 시스템의 관리와 회선의 유지를 위해 막대한 비용을 지불하고 있는 실정이며 전자상거래 시스템에 대한 로드와 트래픽의 증가는 고객에 대한 응답 속도의 저하를 가져와 e-비지니스 고객에 대한 고객만족도를 떨어뜨리는 요인이 되고 있다. 따라서 전자상거래 시스템의 효율적인 관리와 응답속도를 고려한 전자상거래 시스템에 대한 연구가 필요하다. 인터넷상에서 사용자의 응답속도에 영향을 미치는 요소는 객체의 크기, 지리적인 위치와 네트워크 트래픽의 상태, 서버의 성능이 있다. 트래픽의 상태나 서버의 성능과 같은 물리적인 요인의 개선에는 상대적으로 많은 비용이 소요된다. 따라서 개별 시스템 자체의 성능향상 보다는 로드와 트래픽의 분산[3],[5],[11]을 통한 성능을 향상을 위해 많은 연구가 진행되고 있다. 전자상거래 시스템 자체에 대한 성능 개선을 통한 응답 속도의 개선은 네트워크 트래픽의 상태와 서버의 성능 개선을 통해 이루어질 수 있으며 이를 위해서는 다음과 같은 두 가지를 고려한 전자상거래 시스템의 설계 및 개발이 필요하다. 첫 번째가 시스템의 로드를 분산시키는 것이다. 이는 여러 개의 복제 서버를 통한 시스템 전체를 구성하는 서버들간의 로드를 분산시키는 방법으로, 사용자 측면에서의 로드를 분산시키는 것과 서버 측면에서의 로드를 분산시키는 방법이 있다. 이를 위해서는 지역서버를 이용한 계층적 구조의 전자상거래 시스템이 필요하며, 이는 지리적 위치로 인한 응답 속도의 저하를 개선할 수 있다. 두 번째는 인터넷 캐싱[6],[7]을 활용하는 방법이다. 캐싱은 동일한 객체에 대한 중복된 요청이 네트워크 대역폭의 상당 부분을 차지하는 불필요한 시스템 자원의 낭비를 줄일 수 있는 효율적인 방법이다. 캐싱은 인터넷 서버, 프락시 서버, 클라이언트 브라우저와 같은 네트워크의 여러 지점에서 구현할 수 있다. 이 경우 캐싱되는 위치는 다르지만 캐싱 기법은 결국 어떻게 한정된 인터넷 캐시의 공간을 효율적으로 사용할 것인가에 대한 공통적인 문제로 접근을 하고 있다. 캐싱을 통해 서버의 부하와 전체 트래픽을 감소시키며 인터넷 사용자에게는 캐싱된 문서를 통해 빠른 응답 시간을 제공할 수 있게된다. 또한 점차 동영상 자료와 같이 용량이 큰 객체를 많이 포함한 전자상거래 사이트가 증가하고 있으며, 이는 객체의 크기로 인한 지연요인을 증가시킬 것이며 이로 인해 웹 캐싱의 관련된 연구가 활발히 진행 중이다. 또한 실제 산업의 현장에서 아직까지 완전한 .NET 환경[4],[5]을 기반으로 한 시스템의 개발은 초기 단계에 머물고 있다. 하지만 .NET의 막강한 성능과 마이크로소프트사의 지원을 통해 향후 서버 시스템 시장에서 우위를 차지 할 수 있을 것이다.

본 연구에서는 마이크로소프트사의 최신 버전인 닷넷을 기반으로 한 전자상거래 시스템에서, 웹 고객의 응답 지연 속도를 개선한 전자상거래 시스템을 제안하였고 실험을 통해 본 시스템의 성능을 분석하였다. 실험 결과 제안한 시스템은 웹 고객의 다양한 요구에 대해 개선된 응답 속도와 빠른 적응성을 가질 수 있으며, 특히 전자상거래 시스템에 대해 동시에 발생하는 다수의 다중 사용자의 요구로 인해 발생되는 응답 속도의 지연을 막을 수 있을 것으로 기대된다.

II. 관련 연구

1. .NET 기술

.NET 프레임워크는 .NET의 가장 핵심적인 구성요소로서 XML 웹 서비스 및 응용 프로그램을 개발, 배포 및 실행할 수 있도록 해주는 환경이다. 그래서 .NET 플랫폼의 다양한 클라이언트 기기에 사용될 모든 운영체제에 기본적으로 포함된다. 이러한 .NET 프레임워크는 개발자에게 기존의 개발환경에 비해 많은 장점들을 제공해 준다. .NET 프레임워크의 일부분인 공통 언어 런타임(CLR)을 통해서 개발자는 자신에게 익숙한 언어로 웹 서비스 및 응용 프로그램을 개발할 수 있고 동일한 실행 성능을 보장받을 수 있다. 또한 다양한 기능이 제공되어 개발자들이 해결해야 하는 문제에 집중 투자할 수 있다. .NET 프레임워크의 구조는 그림 1과 같다.

.NET 프레임워크를 구성하는 요소 중 가장 중요한 요소는 공통 언어 런타임(CLR:Common Language Runtime)과 .NET 프레임워크 클래스 라이브러리(.NET Framework class library)라 할 수 있다.

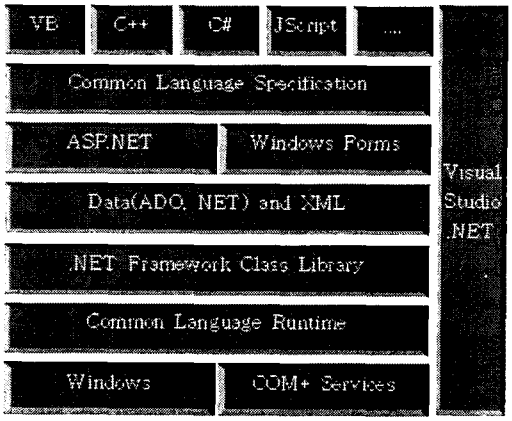

그림 1. 닷넷프레임워크

이 두 요소는 일관성 있는 객체지향 프로그램 환경, 코드 실행 환경의 개선, 업계 표준을 통해 다른 코드와 .NET 프레임워크에서 실행되는 코드와의 원활한 통신 등을 목적으로 .NET 프레임워크에서 도입한 개념이다.

1) 공용 언어 런타임

공용 언어 런타임(CLR:Common Language Runtime)은 그림 2와 같이 .NET 프레임워크의 기초로 .NET 프레임워크에서 XML 웹 서비스 및 응용프로그램을 실행시키는 위해 제공하는 공통 실행 환경이다. .NET 이전의 개발 언어들은 서로 각각 다른 런타임을 사용해왔기 때문에 개발 언어마다 다른 데이터형을 사용한다든지 하여 호환성에 문제가 있었다. .NET은 이런 문제를 해결하여 하나의 통합된 런타임을 제공하며 이를 공통 언어 런타임이라고 한다. 이 공용 언어 런타임은 Java의 가상머신(JVM:Java Virtual Machine)과 유사한 개념으로서 기종에 관계없이 지원되는 공용 언어 런타임만 있다면 .NET 응용프로그램을 실행할 수 있다. 공용 언어 런타임이나 JVM 환경 모두 두 소스 코드가 실행될 하드웨어에 맞도록 컴파일되는 것이 아니라 런타임을 기준으로 컴파일되기 때문에 하드웨어에 대한 의존성을 줄이고 다양한 환경에서도 같은 코드를 실행할 수는 있는 것이다. 이러한 장점을 가진 공용 언어 런타임은 실행되는 응용 프로그램에 대하여 다음과 같은 기능들을 제공한다.

- 응용 프로그램의 코드 관리
- 메모리 관리
- 쓰레드 관리
- 원격 호출 관리
- 시스템이나 다른 응용프로그램에 문제 발생여부 검사
- 발생할 수 있는 문제에 대한 미연 방지

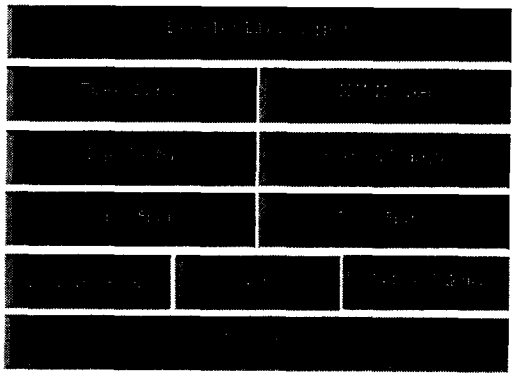

그림 2. 닷넷의 CLR 구조

2) .NET 서버

.NET에서의 서버란 Microsoft .NET Enterprise 서버를 말하는데, 이것은 특정 단일 서버 제품을 지칭하는 것이 아니라 Windows 2000 서버군을 포함한 마이크로소프트의 여러 서버 제품군들을 나타낸다. Microsoft .NET Enterprise 서버 제품군들은 XML 웹 서비스 배포, 관리 및 조정를 위해 개발되었으며, 참고적으로 그 구성을 살펴보면 Windows 2000 서버 외에 표 1과 같은 서버들이 포함되어 있다.

표 1. .NET SERVER

서 버	설 명
Application Center 2000	고가용도 및 확장성 웹 응용 프로그램을 배치하고 관리
BizTalk Server 2000	응용 프로그램 및 조직 전체의 XML기반 비즈니스 프로세스 구축
Commerce Server 2000	확장 가능한 전자상거래 솔루션 구축
Content Management Server 2001	동적 e-비즈니스 웹 사이트의 컨텐트 관리
Exchange Server 2000	메시지 전송과 협력
Host Integration Server 2000	레거시 시스템에서 데이터와 응용 프로그램의 다리 역할
Internet Security and Acceleration Server 2000	안전하며 빠른 인터넷 연결을 제공
Mobile Information 2001 Server	이동용 장비에서 애플리케이션을 사용할 수 있도록 함
SharePoint Portal Server 2001	비즈니스 정보의 검색, 공유 및 게시
SQL Server 2000	구조화된 XML 데이터의 저장, 검색 및 분석

이들 제품군들은 각각의 설명에서 볼 수 있는 바와 같이 웹서비스, XML 데이터 처리, 컨텐츠 관리, DB 처리 등의 기업 업무를 위한 핵심적인 작업을 수행하며, 이동전화나 PDA 등의 Mobile 장치에서도 응용프로그램들을 사용할 있게 해준다.

3) .NET에서 관리형 코드의 컴파일과 실행 과정

.NET에서 관리형 코드의 컴파일과 실행 과정, 즉 공용 언어 런타임 환경 하에서 응용 프로그램의 실행 코드가 만들어지고 실행되는 과정을 살펴보면 다음 그림 3과 같다.

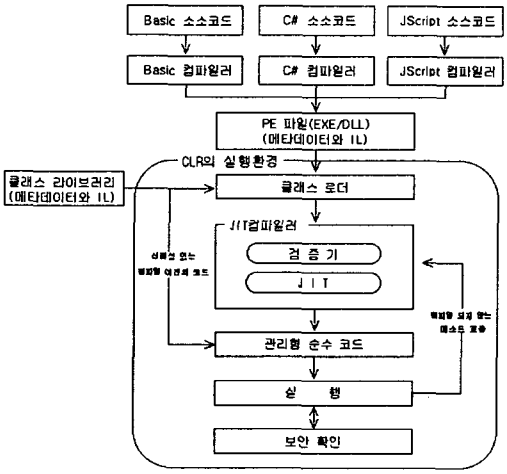

그림 3. .NET 관리형 코드 컴파일과 실행

그림 3의 과정은 크게 소스 코드의 해당 언어 컴파일러로 PE파일 생성하는 단계와 CLR의 JIT 컴파일러에 의한 컴파일과 실행 단계로 나눌 수 있다.

- PE(Potable Executable) 파일 생성 단계: 각각의 프로그램 언어로 작성된 코드는 해당 언어의 컴파일러에 의해 메타데이터와 IL 코드로 구성된 PE 파일로 컴파일된다. 각 컴파일러가 해당 언어의 소스 코드를 컴파일하면 소스 코드는 IL 코드로 변환되고 여기에 메타데이터를 추가해서 EXE나 DLL 파일을 생성한다.
- JIT 컴파일러에 의한 컴파일과 실행 단계: IL 코드로 작성된 PE 파일은 인터프리터 같은 CLR의 JIT(Just In Time) 컴파일러를 통해서 시스템의 CPU가 이해할 수 있는 원시 코드(Native Code)로 변환되어 실행된다. 그러므로 해당 운영 체제 시스템에 맞는 JIT 컴파일러를 채택함으로써 다른 기종에서도 같은 IL 코드의 PE 파일이 실행 가능하다.

2. 부하 분산 연구

병렬 처리 기법의 한계를 뛰어넘어 시스템의 성능을 향상시키기 위한 방법으로 네트워크에 접속된 여러 개의 서버를 하나로 연결하여 부하를 분산시키는 클러스터 기법이 대두되고 있으며 윈드와이드웹서비스 기반하의 클러스터 기법을 웹 클러스터(Web Cluster)라 한다. 클러스트 부하 분산 기법은 크게 사용자 측면에서의 부하 분산과, 서버 측면에서의 부하 분산 기법으로 나누어 볼 수 있다. 사용자 측면에서의 부하 분산은 사용자가 클러스터링된 웹서버들 중 부하가 상대적으로 적게 걸리는 서버를 동적으로 선택할 수 있도록 하는 방법을 사용하지만 사용자의 응용프로

표 2. 기존의 인터넷 캐시를 위한 대체 알고리즘

종류	특징	장점 및 단점
Broadcasting (Kangasharju and Ross, 1999)	물리적으로 네트워크에 도달한 트래픽을 모든 서버에 전달시키고 그 중 특정한 한 서버만이 응답을 하는 방식으로, 이더넷(Ethernet)의 브로드캐스팅 특성을 이용	- 병목현상을 발생시키는 단일 지점이 없음 - 서버 운영체제의 이더넷 어댑터 수정이 필요 - GNU/Linux와 같이 소스코드가 공개되고 수정이 허용되는 운영체제 이외에는 적용될 수 없음 - 불필요한 트래픽이 증가 가능
Round-robin DNS (Kangasharju and Ross, 2000)	사용자가 해당 서비스의 도메인 주소(Domain Address)를 DNS 서버를 통해 IP주소로 변환하는 과정에서 변환을 요청할 때마다 여러 서버의 다른 IP를 알려 주어 부하를 분산	- 서버의 운영체제에 비종속적 - 적은 기회 비용으로 손쉽게 구축 - 연결 지연(Delay)이 존재 - 페일 세이프 기능이 구현되지 못함
Direct Routing (Kim et al., 2000)	- 요청을 중계해 주는 서버가 존재하여 클라이언트의 요청을 서비스 서버에 연결 - 클라이언트의 최초 요청은 중계 서버를 거쳐 서버에 연결이 되지만 요청에 대한 서버의 응답은 클라이언트와 직접 통신	- 중계 서버의 트래픽 집중화를 막음 - 트래픽의 효율적 분배가능 - 서버의 운영체제에 제약이 있음 - 서버의 트래픽 처리량을 정확하게 계산할 수 없음

그램을 바꾸어야 한다는 점과 네트워크 부하를 증가시킨다. 서버 측면에서의 부하 분산은 사용기법에 따라 RR-DNS(Round Robin Domain Name System), 응용계층 스케줄링 기법, IP 계층 스케줄링 기법으로 나누어진다[11].

3. 웹 캐싱

캐싱은 인기있는 서버의 부하와 전체 네트워크의 트래픽을 감소시키며 인터넷 사용자에게는 캐싱된 문서의 빠른 응답 시간을 제공한다[4],[5]. 대체 알고리즘의 성능은 캐싱 기법의 중요한 성능요소가 되며 이를 위해 FIFO(First In First Out), LRU(Least Recently Used), LFU(Least Frequently Used), 그리고 SIZE와 같은 여러 대체 알고리즘들이 연구되어 왔다[4],[5].

인터넷 캐싱에서의 대체 기법은 파일 시스템, 가상 메모리 시스템과 같은 전통적인 기법과는 다르다. 인터넷 캐싱에서는 가변 크기의 인터넷 객체를 지원해야 한다. 인터넷 객체의 크기는 수 바이트에서 수십 메가 바이트까지 매우 다양하며 각 객체의 인기도 또한 매우 가변적이다[6]. 경우에 따라서는 인기가 없는 매우 큰 객체의 삽입이 크기는 작으나 매우 인기 있는 수많은 객체들이 한꺼번에 제거할 수도 있다. 이 경우 인터넷 캐싱의 효율은 매우 떨어지게 된다. 인터넷 캐싱 시스템의 객체들은 클라이언트 브라우저, 프락시 서버, 그리고 원본 서버에 캐시될 수 있다. 클라이언트는 요청한 문서가 브라우저의 캐시에 존재하지 않으면 프락시 서버에 요청한다. 클라이언트로부터 요청을 받은 프락시 서버는 요청된 객체가 캐시되어 있는지를 검사하여 존재할 경우 클라이언트에 해당 서비스를 제공하고 존재하지 않을 경우 프락시 서버는 원본 서버에 요구하여 전송받은 객체를 자신의 캐시 공간에 저장한 후 클라이언트에 제공하게 된다[7].

인터넷 캐싱 기법의 문제는 캐싱되는 위치는 다르지만 어떻게 한정된 저장공간을 효과적으로 관리하는가에 달려있으며 캐싱에 사용되는 시스템의 성능과 지리적인 위치는 대체 알고리즘의 성능과 함께 사용자의 지연 속도를 줄일 수 있는 중요한 요인이다. 또한 캐시서버로만 동작하는 별도의 시스템 설치를 통해 사용자의 응답속도를 개선할 수 있는 캐시전용서버의 설치로 응답지연시간의 개선을 가질 수 있다. 캐시전용서버를 사용하는 경우 이에 사용되는 효율적인 대체 알고리즘의 연구가 필요하게 된다. 또한 인터넷상에서 사용자의 응답속도에 영향을 미치는 요소는 객체의 크기, 지리적인 위치와 네트워크 트래픽의 상태, 서버의 성능이 있다. 트래픽의 상태나 서버 자체 성능의 향상을 통한 물리적인 요인의 개선은 상대적으로 많은 경비가 요구된다.

III. 시스템의 구성

1. 시스템 고려 사항

웹자료들은 점차 멀티미디어화 되어 가고 있으며 시스템의 구성에서 다음과 같은 요소를 고려해야한다.

- 네트워크의 제한된 대역폭 문제
- 고객의 정보 요청 시에 발생하는 지연 시간의 문제: 특히 동시에 집중적으로 발생하는 다수의 사용자에 대해 더욱 심각한 문제가 되고 있다.
- 신뢰성과 안정성의 문제: 오류가 없는 정보의 교환을 위해서는 상대적으로 네트워크 상에서의 부가적인 오버헤드가 발생하며 이는 지연시간의 증가를 가져와 시스템의 성능을 저하시키는 요인이 되고 있다.

이러한 점들을 고려하면 전자상거래 시스템은 안정적이고 고객의 요구에 대해 발생할 수 있는 지연 시간을 최소화할 수 있어야한다. 그림 4는 시스템의 실험을 위한 프로토타입 모델을 나타낸 것이다.

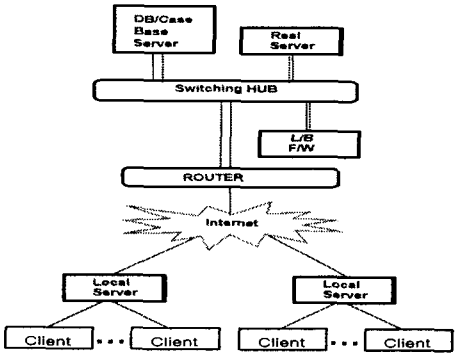

그림 4. 실험 시스템

2. 로컬서버

시스템 자체에서 발생하는 로드는 먼저 로드 밸런싱을 이용하여 다수로 구성된 서버들을 통해 로드를 분산하게 되며, 다수의 로컬서버로 구성된 복제 서버를 통해 시스템 전체의 로드와 네트워크 트래픽을 분산시키게 된다. 본 시스템은 다수의 지역서버로 구성된 계층적 구조의 전자상거

래 시스템이며 트래픽의 분산과 지리적 요인으로 인한 응답 속도의 저하를 개선할 수 있다. 로컬서버는 또한 인터넷 캐싱 기능의 제공을 통해 서비스의 지연 시간을 줄일 수 있다. 캐싱은 인터넷 서버, 프락시 서버, 클라이언트 브라우저와 같은 네트워크의 여러 지점에서 구현할 수 있으며 이 경우 캐싱되는 위치는 다르지만 캐싱 기법은 결국 어떻게 한정된 인터넷 캐시의 공간을 효율적으로 사용할 것인가에 대한 공통적인 문제로 접근을 하고 있다. 캐싱을 통해 서버의 부하와 전체 트래픽을 감소시키며 인터넷 사용자에게는 캐싱된 문서를 통해 빠른 응답 시간을 제공할 수 있게 된다.

프로토타입에서는 로드를 분산시키기 위해 1차적으로 메인 웹서버 시스템 자체에서 다수의 서버로 구성된 구조를 가지며 2차적으로 로컬서버를 이용한 2계층 구조를 통해 로드를 분산시키게 된다. 웹서버 시스템 자체에 대해서는 로드 밸런싱 기능을 통해 로드의 분산이 가능하며 로컬서버는 2계층 기능을 담당하며 트래픽의 양과 사용자의 수에 따라 다수 개의 서버로 구성이 가능하다.

3. 웹캐싱

전자상거래 시스템에서 동영상을 포함한 자료가 점차 늘어나고 있으며 이에 따라 효율적인 웹캐싱을 위한 방법이 절실히 요구되고 있다. 인터넷 객체의 캐싱은 공간 지역성을 이용할 수 없다. 인터넷에서는 각각 객체 단위로 캐싱됨으로 이러한 공간 지역성에 의한 캐시의 적중은 기존의 파일 시스템과는 달리 발생하지 않으며 인터넷 캐싱은 전적으로 같은 문서에 대해 시차를 두고 발생하는 시간 지역성에 따르게 된다.

분할된 캐시를 사용할 경우 캐시에서의 분할된 캐시영역의 수와 각 분할된 캐시영역에 할당하는 영역의 크기, 또한 객체를 지원하는 크기의 설정은 본 시스템의 성능에 영향을 주는 중요한 요소이다. 캐시의 적중률을 높이기 위해서는 자주 사용되는 객체의 크기가 속한 영역의 캐시의 크기를 그렇지 않은 캐시의 영역에 비해 크게 할당하여야 한다. 본 시스템에서는 캐시 영역의 분할을 위해 로그분석을 통해 인터넷 객체의 특성을 분석하였고, 사용자가 요청한 객체의 크기의 비율에 대한 산술적인 값을 구하여 그 결과값을 활용하였다.

로그 분석 결과 100B에서 100KB의 객체에 대한 요청 횟수가 가장 빈번히 일어났으며, 그 중에서 1KB에서 10KB까지의 객체에 대한 요청이 가장 많았으며 첫 번째, 크기가 큰 객체(10KB 이상)보다 작은 크기의 객체(10KB 이하)에 대한 요청 횟수가 빈번하였고 두 번째, 크기가 큰 객체에 대한 적은 횟수의 요청이 한꺼번에 네트워크의 트래픽을 증가시켰다.

본 시스템에서는 캐시 영역을 10KB 이상의 객체에 대한 LARGE와 10KB 이하의 객체에 대한 SMALL 두 개의 영역으로 분할하였으며 각기 다른 크기의 분할된 캐시 영역을 사용한 분할된 캐시 영역을 기반으로 한 캐시 알고리즘을 사용한다.

캐시 관리자는 캐시에 저장된 인터넷 객체의 리스트를 관리한다. 클라이언트의 객체 요청이 들어오면 캐시 관리자는 요구된 객체의 크기에 따라 분류하여 해당 캐시 영역에 존재하는 지를 확인한다. 이때 클라이언트에서 요청된 객체가 캐시 영역에 존재하여 적중이 될 경우 해당 클라이언트에 서비스를 제공하게 되며 캐시 관리자는 이용된 시간 기록을 저장하여 LRU에 의한 객체의 대체에서 높은 순위를 할당받을 수 있게 한다. 만일 적중하지 않으면 관리자는 해당 인터넷 서버에 서비스를 요청하여 해당 객체를 전송 받는다. 전송 받은 객체는 크기에 따라 해당 등급을 분류 받은 후 캐시 관리자는 해당 등급의 캐시 영역에 이 객체를 저장할 공간이 있는지를 확인한다. 저장할 공간이 있을 경우 객체를 캐시에 저장하게 되고, 없을 경우 해당 캐시 영역에 LRU에 의해 여유 공간을 배정하고 객체를 저장한다. 이와 같이 본 시스템에서는 인터넷 객체의 크기를 고려하여 나누어진 캐시 영역 기반의 대체기법을 사용한다. 분리된 캐싱 기법의 원리는 참조의 지역성을 유지하며 각 분할된 부분 캐시의 적중률을 높이는 방법이다. 각 부분에 저장된 인터넷 객체는 같은 등급의 객체들간에만 대체된다.

IV. 실험 및 고찰

실험은 .net 환경에서 asp.net을 기반으로 평균 객체 적중률의 측정을 통해 성능을 평가하였다. 평균 객체 적중률은 클라이언트에서 요청한 전체 객체의 크기에서 적중한 객체 크기 양의 비율로 구해진다. 웹캐싱에서 평균 객체 적중률이 높다는 것은 클라이언트의 요청에 대해 좀더 빠른 응답을 제공할 수 있게 하며, 메인 시스템의 로드를 효율적으로 줄일 수 있게 한다. 또한 메인 시스템과 로컬서버 시스템과의 트래픽을 감소시킬 수 있다. 일반적으로 인터넷을 기반으로 한 전자상거래 시스템에서는 환경의 특수성으로 인해 응답 시간의 향상은 실험을 통해 예측할 수는 있으나 정확

하게 응답시간을 예측하는 것은 어렵다. 이것은 전자상거래 시스템에 대한 인터넷상의 응답 속도는 객체의 크기뿐만 아니라 지리적인 위치와 환경에 따른 네트워크 상황과 시간적이고 공간적인 네트워크의 트래픽 상황에 따라 크게 영향을 받기 때문이다. 본 시스템에서는 웹캐시 기능을 가진 로컬서버 시스템의 구성을 통해 지리적이고 공간적인 영향에 의한 응답 속도의 저하를 줄일 수 있다.

일반적으로 대체 기법의 성능 평가의 척도로는 캐시 적중률(cache hit ratio), 바이트 적중률(byte hit ratio), 지연 감소률(delay saving ratio), 비용 절감률(cost saving ratio) 및 응답 시간 등이 사용되고 있다. 본 실험에서는 클라이언트에서 요청한 객체에 대한 캐시 영역에서의 객체의 존재 여부 파악을 통한 객체의 평균 적중률과 사용자의 요구에 대한 객체의 응답 시간의 평가를 통해 캐싱 기법의 성능을 평가한다.

본 실험에서는 평균객체적중률을 통해 성능을 평가하였다. 평균객체적중률은 클라이언트에서 요청한 전체 객체의 크기에서 적중한 객체 크기 양의 비율로 구해진다. 웹 캐싱에서 평균객체적중률이 높다는 것은 클라이언트의 요청에 대해 좀더 빠른 응답을 제공할 수 있게 하며, 시스템의 로드를 효율적으로 줄일 수 있고 시스템의 트래픽을 감소시킬 수 있다.

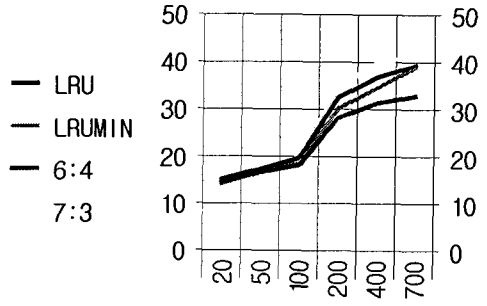

그림 5. mean value of object hit ratio(%)

실험은 캐시의 영역을 LARGE에 60%, SMALL에 40%을 할당한 경우와 캐시 영역 LARGE에 70%, SMALL에 30%을 할당하여 평균 객체 적중률과 클라이언트의 요청에 대한 응답 시간을 측정하였다. 실험결과 그림 5와 같이 LRU나 LRUMIN의 경우 캐시 공간이 늘어날수록 효율이 높아짐을 알 수 있다. 적은 용량의 캐시를 할당한 경우 LRU나 LRUMIN, 6:4, 7:3 기법의 적중률이 거의 차이가 없음을 알 수 있다. 용량이 커짐에 따라 LRU나 LRUMIN 기법에 비해 분할된 캐시방법이 2%-10% 정도의 적중률 향상이 있는 것으로 나타난다.

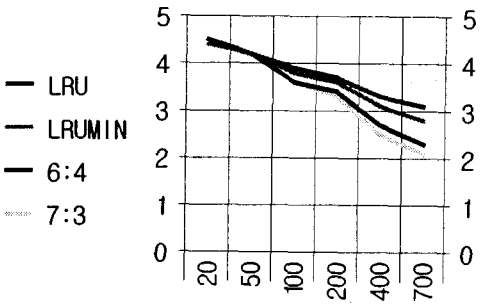

그림 6. mean value of response time(sec.)

본 실험은 트래픽의 여건과 성격에 따라 많은 차이가 있을 것이다. 예를 들어 mp3 파일이나, 각종 멀티미디어 자료의 빈번한 요청과 같은 사용자의 선호도에 따른 특정 객체의 집중적인 요청은 실험 결과에 큰 영향을 미칠 것이다. 사이즈가 큰 객체의 비적중으로 인한 객체의 대체에 소요되는 시간은 사이즈가 작은 객체의 대체에 걸리는 시간에 비해 클라이언트의 응답 시간을 지연시키는 요인이 된다. 따라서 동영상 자료와 같이 용량이 큰 객체를 많이 포함한 경우 6:4보다는 7:3 할당에 의한 캐시의 분할이 좀 더 효율적일 것이며 객체 요청에 따른 캐시의 분할에 대한 추가적인 연구가 필요하다. 그림 6은 클라이언트에서 요구한 객체에 대한 응답 시간을 측정한 것이다. 응답 시간은 캐시 영역이 커짐에 따라 감소하였고, LRUMIN이 LRU 보다는 다소 향상된 결과를 보였고 6:4나 7:3은 적중률과 비슷한 결과를 얻을 수 있었으며 이를 통해 제안 기법의 효용성을 확인할 수 있었다.

V. 결론

인터넷상에서 객체의 크기, 지리적인 위치와 네트워크 트래픽의 상태, 서버의 성능이 응답 속도에 영향을 미치는 요소이다. 이 중에서 트래픽의 상태나 서버 자체 성능 향상을 통한 물리적인 요인의 개선은 많은 비용이 요구된다. 또한 점차 동영상 자료와 같이 용량이 큰 객체를 많이 포함한 전자상거래 사이트가 증가하고 있으며, 이는 객체의 크기로 인한 지연요인을 증가시킬 것이다.

본 연구에서는 마이크로소프트사의 최신 버전인 닷넷을 기반으로 한 전자상거래 시스템에서, 웹 기반 시스템의 효율적인 운용을 위하여 웹 객체의 크기에 대한 참조 특성을 고려한 분할 영역 기반의 웹 캐시 대체 기법을 제안하고 실험을 통해 성능을 분석하였다. 제안 기법은 사용자의 참조 특성의 변화에 대해 유연성을 가지고 있으며 실험결과 기존의 대체 기법인 LRU와 LRUMIN에 비해 객체 적중률과 응답속도의 개선을 확인하였다. 실제 산업의 현장에서 아직까지 완전한 .NET 환경을 기반으로 한 시스템의 개발은 초기 단계에 머물고 있다. 하지만 .NET의 막강한 성능과 마이크로소프트사의 지원을 통해 향후 서버 시스템 시장에서 우위를 차지 할 수 있을 것이다. 따라서 향후 본 연구에서 제안한 시스템을 실제 .NET을 기반으로 한 전자상거래 현장에서의 활용을 통해 시스템 적응성에 대한 검증이 필요하다.

참 고 문 헌

[1] Soe-Tsyr Yuan, A. Liu, "Next-generation Agent -enabled Comparison Shopping," Expert Systems with Applications 18, pp.283-297, 2000.

[2] Elizabeth A. Kendall, "Role Modeling for Agent System Analysis, Design, and Implementation," IEEE Concurrency, Vol.8, No.2, April-June 2000.

[3] V. Cardellini, M. Colajanni, P. S. Yu, "Dynamic Load Balancing on Web-Server Systems," IEEE Internet Computing, pp.28-39, May June 1999.

[4] MicroSoft, technical resources, http://www.microsoft.com/net/.

[5] MicroSoft, What Is .NET, http://www.microsoft.com/net/basics/.

[6] V. Almcida, A. Bestavros, M. Crovella, and A. Oliveira, "Characterizing Reference Locality in the WWW," In Proc. of the 4th Int'l Conf. on Parallel and Distributed Information Systems, 1996.

[7] J. C. Bolot and P. Hoschka, "Performance engineering of the World-Wide Web: Application to dimensioning and cache design," Proc. of the 5th Int'1 WWW Conf., 1996.

[8] D.A Patterson and J. L. Hennessy, "Computer Architecture A Quantitative Approach," Morgan Kaufmann Publishers, 1996.

[9] Chad Yoshikawa, et. als., "Using Smart Clients to Build Scalable Services," http://now.cs.berkeley.edu, USENIX'97, 1997.

[10] Robert L. Carter, Mark E. Crovella, "Dynamic Server Selection Using Bandwidth Probing in Wide Area Net-works," http://www.ncstrl.org, Boston University Technical Report, 1996.

[11] Wensong Zhang, Shiyao Jin, Quanyuan Wu, National Laboratory for Parallel & Distributed Processing, "Creating Linux Virtual Servers," http://proxy.iinchina.net/~wensong/ppfvs/linuxexpo.html

[12] Eric Dean katz, Michelle Butler, and Robert Mcgrath, "A Scalable HTTP Server : The NCCA Prototype," Computer Networks and ISDN Systems, pp.155-163, 1994.

[13] Daniel Anderson, Tao Yang, Veard Holmedahl, and Oscar H. Ibarram "SWEB: Towards a Scalable World Wide Web Server On Multicomputers," Proceedings of International Conference on Parallel Processing, pp.15-19, April, 1996.

[14] Cisco System, "Cisco Local Director," http://www.cisco.com/warp/public/751/lodir/index.html.

[15] Jian Liu, Longlu Xu, Baogen Gu, Jing Zhang, "A Scalable High Performance Internet Cluster Server," High Performance Computing in the Asia-Pacific Region 2000. Proceedings, The Fourth International Conference/Exhibition, Vol.2, pp.941-944, 2000.

나 윤 지(Yun-Ji Na) 정회원

1994년 : 경북대학교 생명공학부 (이학사)

2001년 : 충북대학교 컴퓨터공학 (공학석사)

2001년 ~ 현재 : 충북대학교 컴퓨터공학(박사과정)

1999년 ~ 2000년 : 뉴욕공과대학(NYIT)대학원 Communication ART 전공(석사과정 2학기 수료)

전)대전보건대학 컴퓨터정보처리과 초빙교수

<관심분야> : CBD, 멀티미디어, 전자상거래, 게임, 네트워크