P2P를 이용한 배포 서버의 부하 분산

손 세 일[†] · 이 석 균[††]

요 약

분산 정보 시스템의 온-라인 유지보수를 위해서는 네트워크 내의 참여 노드들에게 파일을 배포하는 작업이 필수적이다. 이때 파일 배포에 대한 사용자들의 요청이 단기간에 집중되면, 배포 서버는 과부하 상태에 빠지며, 이를 플래시 크라우드(Flash Crowds)라 부른다. 플래시 크라우드를 회피하기 위한 일반적인 해결책은 하드웨어의 용량을 증설하는 것이다. 본 논문에서는 추가 비용의 발생 없이 P2P 기반의 소프트웨어적 해결책을 제안한다.

제안된 해결책에서 네트워크의 노드들은 인접한 노드들을 중심으로 서브네트워크들로 구성된다. 각 서브네트워크 내에서 배포 파일의 복사본은 노드들 상호간에 전송될 수 있어 배포 서버의 부하를 분산 시킨다. 효율성을 높이기 위해 배포 대상 파일들은 하나의 패키지로 묶여지고 전송에 앞서 패키지는 동일한 크기를 갖는 다수의 세그먼트들로 분할된다. 정상 상태에서 배포 서버는 노드가 요청한 패키지를 세그먼트 단위로 전송한다. 그러나 배포 서버의 과부하 상태에서 노드가 필요한 세그먼트가 이미 서브네트워크 내에 존재할 경우, 서브네트워크 내의 노드는 필요한 세그먼트를 인접 노드로부터 전송받을 수 있다. 본 논문에서는 이를 처리하기 위한 자료구조와 알고리즘을 제안하고 시뮬레이션을 통해 성능 개선을 확인하였다.

키워드 : P2P, 파일 배포, 부하 분산

Load balancing of a deployment server using P2P

Sei-Il Son[†] · Suk Kyoon Lee[††]

ABSTRACT

To perform on-line maintenance for Distributed Information System, it is indispensable to disseminate files to participant nodes in the network. When users' requests for file deployment occur simultaneously in a short period, a deployment server falls into overload phase, which is often called Flash Crowds. A common solution to avoid Flash Crowds is to increase hardware capacity. In this paper, we propose a software solution based on P2P, which does not cost any additional expense.

In the proposed solution, nodes in the network are grouped into subnetworks one of which is composed of only neighboring nodes. In each subnetwork, copies of deployment files can be transferred to each other. Consequently, it brings about the effect of load balancing in deployment server. To raise the effectiveness, target files for deployment are packed into one package. Before being transferred , each package is divided into multiple equal-sized segments. A deployment server in a normal phase transmits a package requested from nodes in segment units. However a deployment server is overloaded, if segments already exist in the subnetwork, participant nodes in the subnetwork receive necessary segments from neighboring nodes. In this paper, we propose data structures and algorithm for this approach and show performance improvement through simulation.

Key Words : P2P, File Deployment, Load Balancing

1. 서 론

새로운 요구 발생이나 외적인 환경 변화에 분산 정보 시스템이 적응하기 위해서는 유지보수가 필수적이며, 최종적으로 변경된 파일들은 사용자들의 컴퓨터로 배포된다[1-3]. 배포(deployment)란 공유된 프로그램 파일 또는 데이터 파일을 사용자에게 전송하는 것이다.

배포 서버는 대부분의 경우 정상 부하 상태이지만, 일시적으로 많은 사용자들의 요청이 집중되면서 과부하 상태에 빠진다. 이를 플래시 크라우드(Flash Crowds)라 부른다[4]. 예를 들어, 기업의 정보 시스템이 수정되고, 기업 내 사용자들이 이용해야 한다면, 출근 후 사용자가 해당 시스템과 관련된 프로그램을 처음 실행할 때, 변경된 파일들은 배포 서버로부터 사용자 컴퓨터로 전송된다. 따라서 사용자들의 파일 전송 요청이 출근 시간 전후에 집중되게 되면서 서버에 플래시 크라우드가 발생한다.

※ 이 연구는 2004학년도 단국대학교 대학연구비의 지원으로 연구되었음.
† 정 회 원 : 단국대학교 대학원 박사과정 수료
†† 정 회 원 : 단국대학교 컴퓨터과학과 부교수
논문접수 : 2005년 10월 10일, 심사완료 : 2006년 1월 24일

한정된 자원을 가진 서버에 사용자들의 요청이 집중되면 처리량을 초과하게 되고, 이러한 요청들은 큐에서 대기하거나 거절된다. 또한 집중된 네트워크 트래픽의 발생으로 인해 대역폭이 급격히 소모된다. 물론 서버를 증설하고 네트워크 대역폭을 확장하는 것도 하나의 대안이 될 수 있다. 그러나 일시적으로 집중되는 요청들을 처리하기 위해 하드웨어를 추가하는 것은 비용 측면이나 시스템 이용률 측면에서 볼 때 바람직하지 않다.

기존의 배포 방식은 클라이언트/서버 구조의 파일 전송이 갖는 한계 때문에 동시에 많은 수의 사용자들에게 서비스를 제공하지 못하는 문제가 있다. 최근 이 같은 문제를 해결하기 위해 P2P 네트워크에 관한 다양한 연구들[5-10]이 진행되었다. 하지만 이 연구들은 효율적인 데이터의 검색에 초점이 맞추어져 있으며, 주로 해싱 기법을 이용하기 때문에 데이터의 저장 위치와 전송 경로에 대한 제어는 불가능하다. 그 결과 지리적으로 인접한 노드에 데이터가 존재함에도 불구하고, 멀리 떨어진 노드로부터 요청한 데이터를 수신하기 때문에 라우팅 경로가 길어져 과다한 네트워크 트래픽이 발생하고, 전송 시간이 길어진다. 또한, 서버의 부하 분산을 위한 이전의 여러 연구들이 있어왔다[11-13].

본 논문의 목적은 별도의 하드웨어 추가 없이 플래시 크라우드 문제를 해결하고, 배포 성능을 향상시키는 것이다. 제안된 방법은 배포할 파일을 독립적인 전송 경로를 갖는 세그먼트라 불리는 단위로 분할 전송하며, 동일한 서브네트워크에 존재하는 세그먼트의 복사본을 이용하여 서버의 부하를 감소시킨다. 즉, 배포되는 파일을 구성하는 세그먼트들을 다수의 피어들로부터 병렬로 다운로드할 수 있으므로, 단일 서버 기반의 파일 배포에 비해 빠른 다운로딩 속도를 제공한다. 공공기관, 기업, 학교와 같이 노드들이 지리적으로 인접한 경우, IP 주소의 프리픽스(prefix)가 동일한 서브네트워크 상의 노드들 사이에 데이터 전송이 이루어지도록 하였다. 그 결과 전송되는 데이터의 라우팅 경로가 줄어 전체 네트워크 상의 트래픽 발생과 전송 시간이 감소하였다.

2. 관련 연구

파일 전송 시 발생하는 부하 분산과 관련된 연구로는 P2P(Peer-to-Peer), CDN(Contents Delivery Network)과 멀티캐스트(multicast)를 이용한 시스템 등이 있다[5-7, 9, 10, 14-21].

P2P는 인터넷 애플리케이션 중 하나로 참여하는 모든 피어들이 클라이언트와 서버의 역할을 모두 수행한다[5-7, 10]. 예를 들어 파일 공유 P2P 네트워크에 참여하는 피어는 파일의 전송을 요청할 수도 있고, 다른 피어들에게 파일 전송 서비스를 제공할 수도 있다. 따라서 기존의 클라이언트/서버 구조에 비해 확장성이 있다.

P2P 네트워크는 중앙 집중화(centralized) 방식과 분산화(distributed) 방식으로 구분할 수 있다. 중앙 집중화 방식은 Napster와 같이 P2P 네트워크 내에 존재하는 모든 콘텐츠들에 대한 인덱스를 중앙 서버가 관리하고, 사용자 요청을 분석하여 서비스를 제공할 피어를 지정한다[9].

분산화 방식은 구조적(structured) 방식과 비구조적인(unstructured) 방식으로 분류할 수 있다. 구조적인 방식은 파일을 검색하는 요청이 접수되면, 분산 해시 함수를 이용하여 키(key)로 변환하고, 라우팅(routing)할 피어를 결정한다. CAN[5], Chord[9] 등이 대표적인 구조적 P2P 네트워크이다. 비구조적 방식은 Gnutella[9], KaZaA[14]와 같이 파일 검색 요청이 들어오면, 인접한 피어들에게 전달하고, 응답한 피어에게 서비스를 요청한다.

CDN(Contents Delivery Network)은 서버와 클라이언트 사이에 전송할 콘텐츠를 캐시하고 있는 캐시 서버(cache server)를 이용하여, 클라이언트의 요청이 있으면 가까운 위치의 캐시 서버로부터 콘텐츠를 전송받도록 해서 전송 비용을 감소시킨다. CDN은 관리자가 근원 서버(origin server)로부터 캐시 서버로의 콘텐츠 이동 시점, 방법, 상태를 관리할 수 있다[15, 16].

CDN은 여러 개의 모듈로 구성되는데 이들 중 중요한 모듈은 다음과 같다. 첫째 클라이언트의 요청을 받아 콘텐츠를 서비스하는 캐시 서버이다. 둘째, 콘텐츠 전송 및 관리 모듈이다. 콘텐츠 전송 및 관리 모듈을 통해 관리자는 콘텐츠를 여러 서버에 전송하고 시스템 동작을 모니터링 할 수 있다. 셋째, 요청 리다이렉터(request redirector)는 클라이언트가 인접한 서버로 접속할 수 있도록 한다. CDN이 서버에 집중되는 부하를 캐시 서버로 분산시킬 수 있지만, 캐시 서버가 서비스할 수 있는 클라이언트 수가 제한되어 있어 결국 클라이언트의 수가 증가할수록 캐시 서버를 확장해야만 한다.

또 다른 배포 부하 분산에 대한 연구로 애플리케이션 레벨의 멀티캐스트(multicast)가 있다. 이것은 멀티캐스트 지원 장비에 의존하지 않는다는 점에서 많은 연구가 진행되었다[17-20]. 애플리케이션 레벨의 멀티캐스트 시스템에서 노드들은 트리 구조로 연결되어 있으며, 전송된 메시지를 다른 노드로 라우트(route)하거나 유니캐스트(unicast)하여 멀티캐스트와 같은 결과를 얻는다. 하지만 트리-기반 멀티캐스트는 메시지 복제와 분배를 트리를 구성하는 내부 노드들이 처리하는데, 멀티미디어 파일이나 대용량 파일과 같이 높은 대역폭이 필요로 하는 전송의 경우 내부 노드가 트리-기반 멀티캐스트 시스템에서 요구하는 성능과 가용성을 충족하지 못한다. 또한 트리를 구성하는 대부분의 노드들은 리프(leaf) 노드로 자원의 공유나 부하 분산에 도움을 주지 못하는 단점이 있다. 이 같은 문제를 해결하기 위해 M. Castro는 콘텐츠를 스트라이프(stripe)이라 불리는 단위로 분할하고, 각각의 스트라이프를 서로 다른 경로로 멀티캐스트하여 내부 노드의 부하를 줄이는 방법을 제안하였다[21].

본 논문에서는 파일 배포 과정에서 특정 시점에 사용자들의 요청이 집중되어 서버의 과부하 상태가 지속되면서, 원활한 서비스를 제공하지 못하는 플래시 크라우드 문제에 관해 연구하였다. 제안된 방법은 P2P 네트워크 모델 기초로

하여 부하를 분산시키기 때문에 미러 서버나 클러스터를 구축하는 방법과 달리 별도의 하드웨어가 필요하지 않다. 본 논문에서는 이전 P2P 구현들이 파일 공유와 효율적 검색에 초점을 둔 것과 달리 파일 전송에 관심을 두었다. 제안된 방법은 인접한 피어들로 서브네트워크를 구성하고, 서브네트워크 내의 피어들 사이에 직접적인 데이터 전송을 통해 부하 분산을 성취했으며, 전체 네트워크 상의 트래픽 발생이 줄었다. 또한 CDN처럼 관리자가 배포할 파일들을 근원 서버로부터 캐시 서버로 이동시키고, 관리할 필요가 없으며, 애플리케이션 레벨의 멀티캐스트에서 발생하는 중간 노드의 부하 집중이 발생하지 않는 장점이 있다.

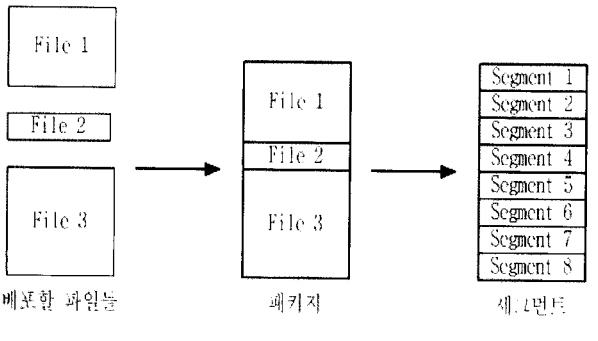
3. 기본 모델

이 장에서는 별도의 하드웨어 추가 없이 특정 시점에 집중되는 서버의 부하를 분산시키기 위해 독립적인 전송 경로를 갖는 세그먼트라 불리는 단위로 패키지를 분할 전송하고, 서브네트워크에 이미 존재하는 세그먼트의 복사본을 이용하여 서버 부하와 네트워크 트래픽, 대역폭 소모를 감소시키기 위해 제안된 배포 방법에 대해 살펴본다.

본 논문에서 서버와 피어(클라이언트)는 네트워크를 구성하는 노드이고, 서브네트워크(subnetwork)란 IP 주소의 프리픽스가 동일한, 즉, 지리적으로 인접한 노드들의 집합을 말한다.

본 논문에서는 서버로부터 피어들에게 배포될 하나 이상의 파일이 존재할 때, 파일들은 배포를 위해 하나로 통합하는데 이것을 패키지(package)라 한다. 하나의 서버에 다수의 패키지들이 존재할 수 있으므로, 각 패키지는 고유의 식별자를 가지며 이것을 pid(package identifier) 부른다. 그리고 패키지가 서버로부터 클라이언트로 전송될 때, 세그먼트(segment)라 불리는 여러 개의 조각들로 분할되며, 각 세그먼트는 고유의 식별자로 sid(segment identifier)를 갖는다. (그림 1)은 파일, 패키지, 세그먼트와의 관계를 보여준다.

배포할 패키지를 분할 전송하지 않는다면, 해당 패키지가 완전히 전송되기 전까지 피어는 클라이언트의 역할만 수행할 뿐 서버의 역할을 할 수 없다. 하지만 패키지를 분할 전송하면, 피어는 수신이 완료된 패키지의 조각, 즉, 세그먼트



(그림 1) 배포할 파일들, 패키지, 세그먼트의 관계

를 이용하여 다른 피어들의 요청을 서비스하는 서버의 역할을 수행할 수 있다. 결국 분할 전송 방법은 짧은 시간 내에 서버 역할을 하는 피어들의 수를 증가시킴으로써 부하 분산과 응답시간을 줄이는 장점을 가진다.

(그림 1)과 같이 임의의 패키지 P가 n개의 세그먼트들로 분할되면, 패키지 P와 세그먼트 S_i와의 관계는 식(1)과 같이 나타낼 수 있다. 분할된 세그먼트의 크기는 마지막 세그먼트를 제외하고는 모두 동일하므로 세그먼트 i의 크기 V_i는 식(2)와 같다.

$$P = S_1 \cup S_2 \cdot \cdot \cdot \cup S_n \tag{1}$$

$$V_1 = V_2 = \cdot \cdot \cdot = V_{n-1} \geq V_n \tag{2}$$

본 논문에서 제안된 배포 방법은 피어의 요청이 있으면 서버가 패키지를 피어로 전송한다. 하나의 세그먼트 전송이 완료되면 서버는 피어가 속한 서브네트워크에 해당 세그먼트가 성공적으로 전송되었음을 서버가 저장하고 있는 SNB(SubNetwork Bit string)에 기록한다. 제안된 방법에서 n개의 세그먼트로 분할된 패키지를 관리하기 위해, $< b_1, b_2, \cdot \cdot \cdot, b_n >$과 같이 표현되는 두 개의 비트 스트링 SNB와 PB(Package Bit string)를 이용한다.

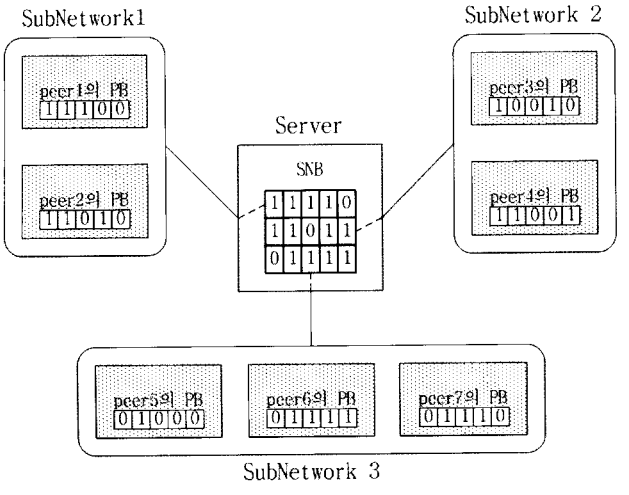
SNB는 서버에 저장되며, 동일한 서브네트워크에 속한 피어들이 임의의 패키지 l을 구성하는 n개의 세그먼트들 중 어떤 것을 수신한 적이 있는지를 보여준다. 예를 들어,서브네트워크 i의 초기 SNB_i의 모든 비트들은 0이다. 만약 서브네트워크 i에 속한 임의의 피어가 sid가 j인 세그먼트를 수신 완료하면, SNB_i의 j번째 비트의 값이 1로 설정된다.

PB는 피어에 저장되며, 피어가 하나의 패키지를 구성하는 세그먼트들 중 어떤 것을 수신했는지를 보여준다. 예를 들어, 패키지 l의 초기 PB_l의 모든 비트들은 0이다. 만약 피어가 sid가 j인 세그먼트를 수신 완료하면, PB_l의 j번째 비트의 값이 1로 설정된다.

(그림 2)는 서버에 다섯 개의 세그먼트들로 분할되는 패키지가 존재하며, 세 개의 서브네트워크에 속한 피어들에게 패키지를 배포할 때의 SNB와 PB를 보여준다. (그림 2)에서 서버에 저장된 SNB의 개수는 패키지를 요청한 피어가 속한 서브네트워크 수와 같은 세 개이다. 각 SNB는 SubNetwork 1, SubNetwork 2, SubNetwork 3에 성공적으로 전송된 세그먼트들이 무엇인지를 보여준다. SubNetwork 1에 대응되는 SNB를 보면, SNB가 <1, 1, 1, 1, 0> 이기 때문에 sid가 1, 2, 3, 4 인 총 네 개의 세그먼트가 SubNetwork 1에 이미 전송되었음을 알 수 있다.

Peer 2의 PB를 보면, PB가 <1, 1, 0, 1, 0> 이기 때문에 세그먼트 ID가 1, 2, 4 인 세 개의 세그먼트를 성공적으로 수신했음을 알 수 있다.

따라서 SNB와 PB의 관계는 식(3)과 같이 정리할 수 있다. 서브네트워크 i에서 m개의 노드들이 존재할 때, 서브네



(그림 2) 다섯 개의 세그먼트들로 분할된 패키지를 배포할 경우 SNB와 PB의 예

트워크 i에 전송된 세그먼트를 나타내는 SNB_i는 서브네트워크 i에 속한 피어들의 PB_{ij}의 논리합과 같다.

$$SNB_i = PB_{i1} \vee PB_{i2} \vee \cdots \vee PB_{im} \quad (3)$$

4. 배포 방법

이 장에서는 서브네트워크에 존재하는 복사본을 이용하여 서버의 부하를 분산시키는 배포 방법에 대해 알아본다.

플래시 크라우드가 발생함으로써 서버의 응답 시간 지연, 서버 다운 현상이 나타나는 것을 예방하기 위해 다음과 같이 서버의 부하 상태를 정상 부하와 과부하 두 개의 상태로 구분한다. 각 상태를 구분하는 기준은 사전에 서버에 설정한 대기 큐의 길이(Q)이다. 현재 대기 중인 요청의 수를 T라 할 때, 정상 부하 상태는 $T < Q$인 경우이며, 과부하는 $T \geq Q$인 경우이다. 서버는 정상부하 상태에서 추가적인 요청에 대해 서비스를 제공할 수 있지만, 과부하 상태에서는 정상부하 상태가 될 때까지 더 이상의 새로운 요청에 대해 서비스를 제공하지 않는다. 제안된 시스템은 기존 프로세스의 부하 분산 방법과는 달리 작업 이전을 하지 않는다.

제안된 방법에서 서버의 처리 단위는 세그먼트가 되며, 서버의 부하 상태가 정상인 경우 피어의 요청을 서버가 직접 처리한다. 이것은 배포 과정의 초기에 서브네트워크 내의 피어들이 세그먼트의 복사본을 충분히 소유하지 못한 경우, 발생할 수 있는 성능 저하를 방지한다.

4.1 피어

피어가 패키지 수신을 위해 사용하는 알고리즘은 다음과 같다. 먼저 수신할 패키지가 있는지 배포 서버에게 질의한다. 서버는 피어가 수신할 패키지가 존재한다면, 현재 서버의 부하량, 패키지 식별자와 함께 대응되는 SNB를 응답으로써 피어에 전달한다. 피어는 수신된 SNB를 보고, 패키지가 몇 개의 세그먼트들로 구성되어 있으며, 피어가 속한 서브네트워크 내에 다른 피어들이 어떤 세그먼트들을 수신한 적이 있는지를 알 수 있다. 피어는 패키지를 구성하는 세그먼트들 중 자신이 소유하지 않는 세그먼트, 즉, PB에서 0으로 표시되는 비트들과 대응되는 세그먼트를 임의로 선택하여 전송을 요청한다.

배포 초기 서브네트워크에는 어떠한 세그먼트의 복사본이 존재하지 않을 수도 있으므로, 서버가 정상 부하인 경우에는 서버에 세그먼트 송신 요청을 한다. 서버가 과부하이고, 서브네트워크에 복사본이 존재하는 경우, 피어는 서브네트워크 내 필요한 세그먼트를 소유한 피어를 검색하기 위해 질의를 브로드캐스팅 하고, 응답한 피어들 중 임의로 하나를 선택하여 세그먼트의 전송을 요청한다. 서버로부터 수신한 SNB를 참조하면, 별도의 질의 없이 이전에 서브네트워크 내로 수신된 세그먼트들이 무엇인지를 알 수 있다.

만약 서버가 과부하 상태이고, 서브네트워크에 해당 세그먼트를 소유한 피어가 없다면, 일정 시간을 대기한 후 서버에 세그먼트 요청을 재시도한다.

피어는 요청할 세그먼트의 선택을 위해 일양 분포(uniform distribution)의 난수를 이용한다. 이것은 서브네트워크 내에 패키지를 구성하는 각 세그먼트들의 수가 편재되는 것을 방지하기 위해서이다. 각 세그먼트들의 수가 균일하지 않으면, 일부 피어들에게 요청이 집중되어 과부하가 발생할 수 있다.

(그림 3)은 피어가 패키지를 수신하는 알고리즘이다. 이때 사용되는 변수와 연산자의 의미는 다음과 같다.

```
Request to server whether package to be downloaded
Get SNB_n. T
if SNB_n = φ exit

While Not Empty ServerOverload(T)
    r ← randomly select segment id from S_p − PB_c
    if T < Q
        request segment r to server
    else
        ans ← broadcate query for r. referenced by SNB_n
        if ans ≠ φ
            c ← first answer client id from ans
            request segment r to client c
            PB_c. SNB_n ← r
        else
            wait for a short time
        end if
    end if
End While
```

(그림 3) 피어가 패키지를 수신하는 알고리즘

S_p : 패키지 p를 구성하는 세그먼트 식별자들의 집합
C : 피어 식별자들의 집합
PB_c : 피어 c가 수신한 세그먼트 식별자들의 집합

SNB_n : 서브네트워크 n에 전송된 세그먼트 식별자들의 집합

T : 현재 서버의 부하량

4.2 서버

서버는 파일들을 하나의 패키지로 통합하고, 이를 세그먼트로 분할하여 피어들에게 배포한다. 또한 서버는 피어가 속한 서브네트워크 별로 전송된 세그먼트들이 무엇인지 SNB를 통해 관리한다. 서브네트워크에 패키지를 구성하는 모든 세그먼트들이 전송되면, 이 서브네트워크에 대응되는 SNB의 모든 비트들은 '1' 값을 갖는다.

서버는 피어로부터 전송된 배포 패키지 존재 여부 질의에 대해 현재 부하량과 SNB로 응답한다. 이것은 서버가 과부하 상태일 때, 피어가 자신의 서브네트워크에 속한 다른 피어들에게 필요한 세그먼트를 검색하는 과정에서 발생할 수 있는 메시지 양을 줄인다.

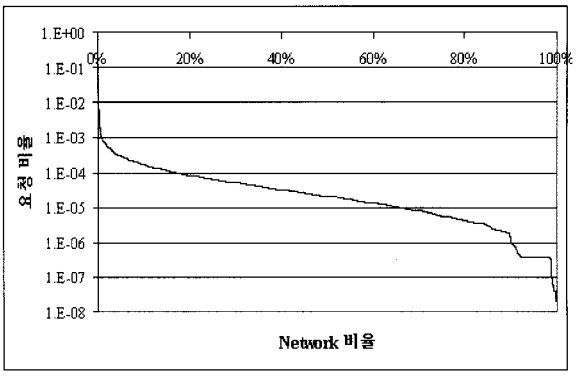
서버는 과부하 상태가 될 때까지 피어들의 세그먼트 전송 요청을 처리한다. 서버가 과부하 상태가 되면, 피어는 서브네트워크 내의 세그먼트 복사본을 이용한다.

서버가 과부하 상태가 될 때까지 피어의 요청을 처리하는 것은 몇 가지 이유가 있다. 첫째, 배포 초기 서브네트워크에 존재하는 세그먼트 복사본의 수가 충분하지 않기 때문에 발생할 수 있는 문제를 예방하기 위해서 세그먼트의 수가 충분해질 때까지 서버가 피어의 요청을 처리하는 것이 바람직하다. 둘째, 서버가 처리할 수 있는 작업을 다른 피어들로 위임하면서 발생하는 부가적인 트래픽과 응답의 지연을 방지한다.

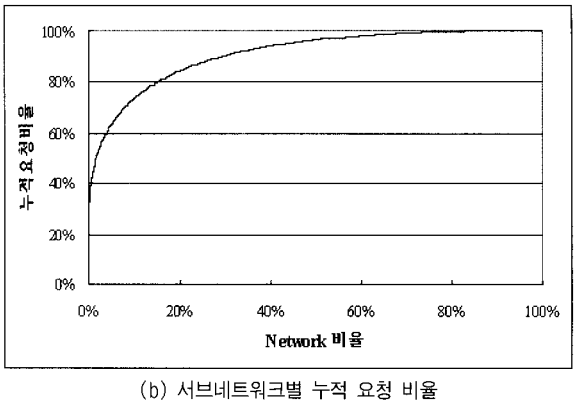
5. 성능 분석

본 장에서는 제안된 방법을 성능 평가를 위해 먼저 서버 A의 2004년 9월 1일부터 2004년 12월 31일까지의 파일 전송 요청 로그를 분석하였다. (그림 4)(a)는 전송 요청을 한 클라이언트가 속한 서브네트워크를 IP주소의 프리픽스를 이용하여 분류한 후, 각각의 서브네트워크별로 전송 요청 비율을 계산하여 *Log*그래프로 나타낸 것이다. (그림 4)(b)는 클라이언트로 전달된 요청들을 서브네트워크별로 구분하고, 이것을 서브네트워크별 요청 비율에 대한 누적 그래프를 그린 것이다. 분석 결과에 따르면, 전송 요청이 많은 상위 10%의 서브네트워크들에 속한 클라이언트들로부터 전체 요청의 78%가 발생되었음을 알 수 있다.

즉, 소수의 서브네트워크들에 속한 클라이언트들로부터 제기된 요청이 서버 부하의 대부분을 차지하고 있다. 따라서 전송 요청을 한 클라이언트가 속한 서브네트워크 내에 요청한 파일의 복사본이 존재하고, 이를 효과적으로 이용할 수 있다면 서버에 집중되는 부하를 상당히 줄일 수 있다. 본 논문에서 제안한 P2P를 이용한 배포 서버의 부하 분산 방법은 이 같은 클라이언트들의 요청 패턴을 고려할 때 의미가 있다.



(a) 서브네트워크별 요청 비율 *Log*그래프

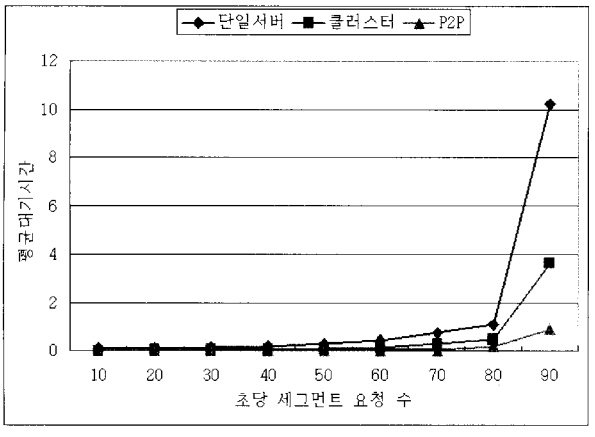


(b) 서브네트워크별 누적 요청 비율

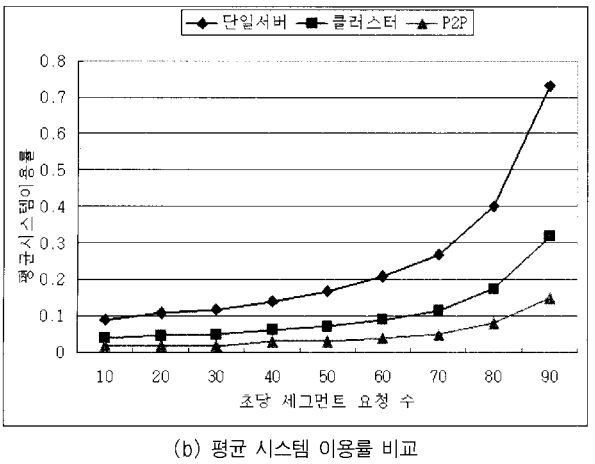
(그림 4) 네트워크별 클라이언트 전송 요청 분석

본 논문에서 제안한 배포 방법과 기존의 단일 서버와 클러스터를 이용한 배포 방식과의 성능 비교를 위해 시뮬레이션을 수행하였다. 성능 비교를 위해 대기시간과 시스템 이용률을 평가 척도로 삼았다. 시뮬레이션 환경은 다음과 같다. 서버는 세 개의 서브네트워크에 속한 피어들의 요청을 처리한다. 각 서브네트워크에는 100개의 피어들이 존재하며, 서버는 총 300개 피어들의 요청을 처리한다. 각 피어는 항상 자신이 소유한 세그먼트의 검색 질의에 응답할 수 있다. P2P 네트워크에 참여하는 피어들은 시뮬레이션 시간 동안 탈퇴(leave)없이 계속 동작한다. 서버와 피어 사이의 통신 비용과 통신 오류에 따른 지체는 고려하지 않았다. 또한, 시뮬레이션에서 피어가 동일한 서브네트워크 내의 다른 피어들에게 필요한 세그먼트의 소유 여부를 브로드캐스팅으로 질의하고, 응답을 수신하는데 필요한 검색 시간은 세그먼트 전송 시간에 비해 매우 작기 때문에 고려하지 않았다.

배포할 패키지의 크기는 50MB이며, 이것은 1MB의 크기를 갖는 50개의 세그먼트로 구성된다. 서버가 피어로부터 수신한 하나의 세그먼트 전송 요청을 처리하는데 소요되는 서비스 시간은 상수값(실험에서는 0.1)을 갖는다. 이것은 패키지를 구성하는 마지막 세그먼트를 제외한 모든 세그먼트의 크기가 1MB로 동일하기 때문에 서비스 시간 역시 동일하다고 가정했기 때문이다. 세그먼트 요청의 발생 분포는 플래시 크라우드의 트래픽 분포에 관한 연구[22]를 참고하여 포아송 분포를 따르는 것으로 하였다. 초당 발생하는 요청



(a) 평균 대기시간 비교



(b) 평균 시스템 이용률 비교

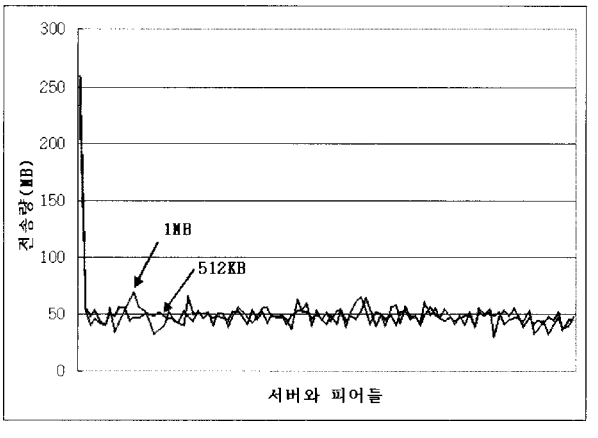
(그림 5) 제안된 방법의 성능 비교

의 수를 10부터 90까지로 10개 단위로 증가시켜가며 실험했으며, 각 조건별로 10회 반복 수행하고, 그 결과의 평균값을 이용하였다. 클러스터를 이용한 방식에서는 복제된 데이터를 소유한 세 개의 서버를 이용하여 부하를 분산시키는 상황을 가정하였다. 클러스터에 참여하는 서버는 단일 서버와 동일한 처리 능력을 가진다.

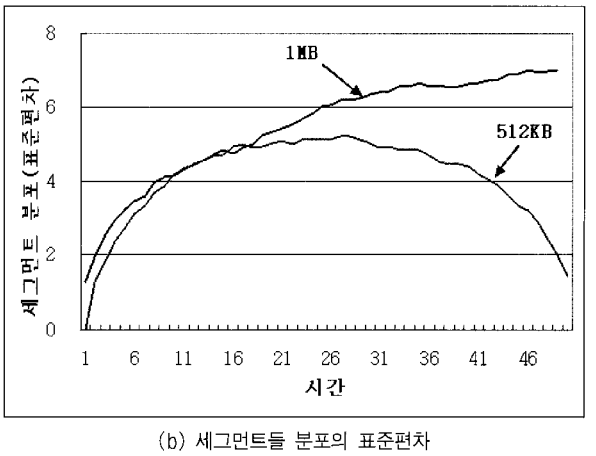
(그림 5)(a)는 제안된 배포 방법과 단일 서버와 클러스터를 이용한 배포 방법의 평균 응답 시간을 보여준다. 피어로부터 서버로 전달되는 초당 세그먼트 요청의 수가 40이하인 경우 배포 방법들 사이의 평균 대기 시간에는 큰 차이가 없다. 초당 세그먼트 요청의 수가 80이상이 되면, 단일 서버와 클러스터를 이용한 배포 방법 모두 평균 대기 시간이 급속히 증가한다. 하지만, 제안된 배포 방법의 평균 대기 시간의 증가는 상대적으로 매우 작다. 이것은 피어가 세그먼트를 전송받자마자 서버의 역할을 하므로 서버의 수가 지수적(exponential)으로 증가하기 때문이다. 따라서 P2P를 이용한 배포 방법은 피어가 소유한 세그먼트들의 복사본을 이용해 다른 피어들의 요청에 대한 서비스를 제공함으로써 서버의 부하를 분산시킨다.

(그림 5)(b)는 제안된 배포 방법과 단일 서버와 클러스터를 이용한 배포 방법에서 초당 요청의 수 증가에 따른 서버의 평균 시스템 이용률 변화를 보여준다. 세 방법 모두 요청의 수가 증가할수록 서버의 이용률이 증가했다. 특히, 단일 서버를 이용하는 경우 초당 요청의 수가 80이상인 경우 시스템 이용률이 급격히 증가함을 하는 것을 알 수 있다. 하지만 제안된 방법은 요청의 수가 증가할수록 서버의 역할을 하는 피어들의 수도 함께 증가해서 P2P 네트워크의 처리 능력이 향상된다. 그 결과 시스템 이용률의 증가폭을 상당히 줄일 수 있다.

(그림 6)은 세그먼트 크기에 따른 서버와 피어들의 특성을 분석하기 위해 서버와 하나의 서브네트워크에 속한 피어들의 세그먼트 전송량과 시간에 따른 세그먼트 분포의 변화를 보여준다. (그림 6)(a)는 세그먼트 크기에 따른 서버와 피어들의 전송량을 보여주며, X축의 가장 왼쪽이 서버를 나머지 오른쪽은 동일한 서브네트워크에 속한 피어를 나타낸다. X축의 값은 피어의 식별자로 본 분석에서는 의미가 없다. Y축은 서버나 각 피어들이 다른 피어들에게 전송한 세그먼트의 양을 나타낸다. 본 논문의 시뮬레이션 결과에 따르면,



(a) 서버와 피어들의 전송량



(b) 세그먼트들 분포의 표준편차

(그림 6) 세그먼트 크기에 따른 특성

(그림 6)(a)와 같이 세그먼트의 크기는 피어가 다른 피어들에게 전송하는 세그먼트의 양과 관계가 없다. 즉, 세그먼트의 크기가 상관없이 피어는 일정한 양의 세그먼트를 다른

피어들에게 서비스한다. (그림 6)(b)는 한 서브네트워크 내에 피어들이 소유한 세그먼트 분포의 표준 편차가 시뮬레이션이 진행됨에 따라 어떻게 변화하는가를 보여준다. 세그먼트 분포의 표준편차가 작다면 서브네트워크 내에 피어들이 소유하고 있는 세그먼트들의 복사본의 수가 일정하기 때문에 피어들 사이에 부하를 고르게 분산시킬 수 있다. 세그먼트 분포의 표준편차가 크다면, 서브네트워크 내에 세그먼트들의 복사본의 수가 편재되어 특정한 세그먼트를 소유한 피어가 상대적으로 많은 전송 요청을 처리하게 된다. (그림 6)(b)를 보면 세그먼트의 크기가 1MB인 경우 시간이 지남에 따라 세그먼트 분포의 표준편차가 커지므로, 특정 세그먼트들을 소유한 피어들에게 상대적으로 많은 부하가 발생하게 된다. 하지만 세그먼트의 크기가 512KB인 경우 일정한 시간동안은 세그먼트들의 분포가 편재되지만 이후 점차 안정된 후 세그먼트들의 복사본의 수가 균일하게 되는 것을 알 수 있다. 따라서 세그먼트의 크기가 작은 것이 부하 분산 측면에서 바람직하다.

7. 결론 및 향후 연구과제

본 논문에서는 P2P 네트워크를 이용하여 파일 배포 시 발생하는 플래시 크라우드 문제 해결에 관해 연구하였다. 제안된 방법은 피어가 서버에 접속한 후 수신할 파일들, 즉 패키지가 존재하는지 여부를 확인하고, 수신할 파일들이 있다면, 서버에게 송신 요청을 한다. 서버가 피어의 요청을 처리할 수 있으면, 패키지는 서버로부터 피어에게 전송된다. 하지만 서버가 과부하 상태라면, 피어는 자신과 지리적으로 인접한 서브네트워크 내의 다른 피어들에게 해당 패키지를 송신해 줄 수 있는지 질의한다. 피어는 제일 먼저 응답한 피어를 선택하여 필요한 세그먼트의 전송을 요구함으로써 파일 배포 과정에서 서버에 집중되는 부하를 분산시켰다. 제안된 방법이 기존의 서버 중심 파일 배포와 비교하여 서버의 부하를 얼마나 분산시키는지를 알아보기 위해 시뮬레이션을 수행했고, 그 결과 상당한 서버 부하의 분산 효과가 있음을 확인하였다. 또한 기존 서버의 파일 요청 로그를 분석하여 전송 요청이 많은 상위 10%의 서브네트워크들의 요청이 전체 요청의 78%를 차지하는 것을 알았다. 이것은 본 논문에서 제안한 방법이 서버의 부하를 효과적으로 분산 시킬 수 있음을 보여준다.

제안된 방법은 P2P 네트워크 모델 기초로 하여 부하를 분산시키기 때문에 미러 서버나 클러스터를 구축하는 방법과 달리 별도의 하드웨어를 요구하지 않는다. 이전 P2P 시스템들과 관련된 연구들은 파일 공유와 효율적 검색에 초점을 두었지만, 본 논문은 검색된 결과, 즉, 전송 과정과 부하 분산에 초점을 두었다. 제안된 방법은 CDN처럼 관리자가 배포할 파일들을 근원 서버로부터 캐시 서버로 이동시키고, 관리할 필요가 없으며, 미러 서버처럼 주기적인 동기화 작업이 요구되지 않는다. 또한, 애플리케이션 레벨의 멀티캐스트에서 발생하는 중간 노드의 부하 집중도 발생하지 않는 장점이 있다.

향후 연구 과제로는 서브네트워크 내의 모든 세그먼트 복사본들의 수를 많은 비용 없이 일정하게 유지하도록 관리하는 방법에 관한 연구가 필요하며, 또한 요청한 세그먼트를 전송할 수 있는 피어가 다수일 때, 어떤 피어를 소스로 선택하는 것이 바람직한지에 대한 연구가 필요하다. 그리고 세그먼트가 서버가 아닌 다른 피어로부터 전송될 때, 세그먼트의 위/변조 여부를 확인할 수 있는 보안 기술에 대한 연구가 필요하다.

참 고 문 헌

[1] F. Berman외 2인, "Grid Computing Making the Global Infrastructure a Reality," WILEY, 2003.

[2] G. Coulouris외 2인, "Distributed Systems Concepts and Design 3rd ed.," Addison-Wesley, 2001.

[3] R. Orfali 외 2인, "Client/Server Survival Guide 3rd ed.," John Wiley & Sons, Inc., 1999.

[4] I. Ari, B.Hong, E. Miller, S. Brandt, D. Long, "Managing Flash Crowds on the Internet", In Proc. of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems, pp.246-249, October, 2003.

[5] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker "A Scalable Content-Addressable Network," In Proc. of ACM SIGCOMM, pp.161-172, August, 2001.

[6] A. Rowstron, P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," IFIP/ACM In Conf. on Distributed Systems Platforms(Middleware), Heidelberg, Germany, pp.329-350, November, 2001.

[7] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," ACM SIGCOMM 2001, San Deigo, CA, pp.149-160, August, 2001.

[8] 이광현, 전형수, 유청중, 장옥배, "서버 부하 감소를 위한 P2P 기반 데이터 서비스 시스템의 설계 및 구현", 정보처리학회논문지C, 제9-C권 제5호, pp.615-626, 2002. 10.

[9] 전형성 외 4인 역, "PEER-TO-PEER 차세대 인터넷 P2P", 한빛미디어, 2001.

[10] E. Pitoura외 4인, "DBGlobe:A Service-Oriented P2P System for Global Computing," SIGMOD Record, Vol.32, No.3, pp. 77-82, 2003. 9.

[11] 이동우, 이성훈, 황종선, "이질형 분산 시스템에서 유전자 알고리즘을 이용한 동적 부하 균등 기법", 정보처리학회논문지A, 제10-A권, pp.49-58, 2003. 3.

[12] 이영, "이기종 웹 클러스터 시스템에 대한 부하분산 알고리즘 연구", 정보처리학회논문지A, 제10-A권, pp.225-230, 2003. 8.

[13] 임경수, 박미희, 김종근, "스타형 분산 컴퓨터 시스템의 동적

부하분산", 공업기술연구논문집, 제21권 제1호, pp.89-96, 1993.

[14] KaZaA file sharing network Home page. http://www.kazza.com

[15] 최승락, 양철웅, 이중식, "CDN의 핵심 구성 기술들과 경향", 한국정보과학회지 제20권 제9호, pp.5-11, 2002. 9.

[16] A. Vakali, G. Pallis, "Content Delivery Networks : Status and Trends," IEEE Internet Computing, Vol.7, No.6, pp.68-74, December, 2003.

[17] S. Banerjee, B. Bhattacharjee, C. Kommareddy. "Scalable application layer multicast," In Proceedings of ACM SIGCOMM, pp.205-217, August, 2002.

[18] K. Birman, M. Hayden, O. OZkasap, Z. Xiao, M. Budiu, and Y. Minsky. "Bimodal multicast," ACM Transactions on Computer Systems, Vol.17, No.2, pp.41-88, May, 1999.

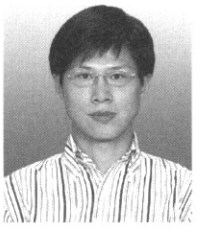
[19] Y. Chu, S. Rao, H. Zhang. "A case for end system multicast," In Proc. of ACM Sigmetrics, pp.1-12, June, 2000.

[20] D. Kostic, A. Rodriguez, J. Albrecht, A. Bhirud, A. Vahdat. "Using random subsets to build scalable network services," In Proc. of the USENIX Symposium on Internet Technologies and Systems, March, 2003.

[21] M. Castro, P. Druschel, A-M. Kermarrec, A. Nandi, A. Rowstron and A. Singh, "SplitStream: High-bandwidth multicast in a cooperative environment," SOSP'03, Lake Bolton, New York, October, 2003.

[22] Ari, B. Hong, E. L. Miller, S. A. Brandt, and D. D. E. Long, "Managing flash crowds on the Internet," MASCOTS'03, pp. 246 - 249, Orlando, FL, 2003.

손 세 일



e-mail : eric31@dku.edu

1993년 유한전문대학 전자계산과

1997년 한국방송통신대학교 전자계산학과(학사)

1999년 단국대학교 대학원 전산통계학과(석사)

2002년 단국대학교 대학원 전산통계학과(박사수료)

1993년~1996년 상지전산(주)

2002년~현재 단국대학교 정보컴퓨터학부 강의전임강사

관심분야 : P2P, 유비쿼터스 컴퓨팅, 트래픽 분석, 데이터베이스, 전자상거래

이 석 균

e-mail : sklee@dankook.ac.kr

1982년 서울대학교 경제학과(학사)

1990년 Computer Science, University of Iowa(석사)

1993년 Computer Science, University of Iowa(박사)

1992년 국제학술대회 ICDE(Int. Conf. On Data Engineering)에서 최우수 논문상 수상

1993년~1997년 세종대학교 정보처리학과 교수

1997년~현재 단국대학교 컴퓨터과학과 부교수

관심분야 : 데이터베이스에서 불완전 정보관리, 데이터베이스 시각 질의어 설계, 다중처리기에서의 실시간 스케줄링, XML, 웹 문서에서의 변화 탐지, 버전닝(versioning) 등