시간 복잡도를 개선한 웹 서버 배치 알고리즘

김 선 호[†] · 윤 미 연[††] · 신 용 태[†††]

요 약

최근 웹 서비스 사용자와 웹상에 대용량 콘텐츠의 급증으로 웹 서버의 부하가 가중되고 서비스의 질이 떨어지는 문제가 발생하였다. 이러한 문제의 해결로 콘텐츠를 다수의 지역 서버에 복제하고 복제 서버로 하여금 클라이언트의 요청을 처리하도록 하여 웹 서버의 부하를 줄이고 트래픽을 분산하는 콘텐츠 분산 네트워크 기술이 사용되고 있다. 그러나, 이러한 다수의 복제 서버 사용하는 환경에서는 복제 서버의 효용을 극대화 할 수 있는 배치 전략이 필요하다. 본 논문에서는 지연과 트래픽에 기반한 각 노드의 비용이 임계 값을 초과하지 않도록 bottom-up 다이나믹 프로그래밍 방법을 이용하여 복제 서버를 배치하도록 함으로써 빠르고 안정적인 서비스가 가능하도록 하였다. 제안하는 알고리즘은 $O(n^2)$보다 작은 $O((n-d^h)\cdot|ch_v|)$의 시간 복잡도로 복제 서버 배치 문제를 해결하였다.

A Replica Placement Algorithm reducing Time Complexity

Seonho Kim[†] · Miyoun Yoon[††] · Yongtae Shin[†††]

ABSTRACT

Recently, contents distribution technologies have been used to cope with the explosive demand for Web services. In this paper, we addressed the issue of the optimal placement of replicas in the environment where Web contents are replicated. We placed replicas so that clients can have access to replicas with the proper delay and bandwidth. We attempted to solve the problem via dynamic programming considering cost of delay and traffic. We have come up with $O((n-d^h)\cdot|ch_v|)$ time complexity that is less than $O(n^2)$. We defined the threshold and proved that our algorithm guarantees the reliable services.

키워드 : 복제 서버(Replicas), 배치(Placement), 시간 복잡도(Time Complexity)

1. 서 론

인터넷 기술의 발달로 웹을 통한 대용량 멀티미디어 서비스가 급증하고 이에 따라 서버와 네트워크의 부담이 가중되고 있다. 웹은 클라이언트-서버 모델을 기본으로 하기 때문에 인기 있는 웹 서버로의 부하 집중은 트래픽의 병목을 발생시키고 사용자의 대기 시간을 증가시키고 있다.

이러한 문제의 해결을 위해 사용자에 의해 자주 요청되는 데이터를 사용자와 가까운 곳에 보관하여 서비스 하는 캐싱(Caching), 특정 사이트에 가중되는 부하를 분산시키기 위하여 사이트의 일부 또는 전부를 복사하여 다른 사이트에 저장하여 운영하는 미러링(Mirroring) 기술이 사용되었으며[1], 최근에는 이러한 기술을 한 단계 발전시켜 원본 서버의 데이터를 사용자와 가까운 위치에 설치된 복제 웹 서버로 복사하여 사용자로 하여금 인터넷 미들 마일을 거치지 않고 바로 가까운 복제 서버로부터 서비스 받도록 하는 CDN(Content Delivery Network)[2]과 서버의 효율적인 자원 분산을 위해 사용자간의 협력으로 자원을 공유하여 사용하고자 하는 P2P (Peer To Peer)[3] 기술이 대두되었다. CDN은 콘텐츠를 원본 서버로부터 인터넷 곳곳에 흩어진 복제 서버에 복사해 놓고 가까이에 있는 복제 서버로 하여금 클라이언트의 요청에 응답하게 하는 서비스이다. 데이터 저장 비용이 네트워크 회선 비용보다 저렴한 현실을 감안할 때 이러한 콘텐츠 분산 기술은 고품질의 인터넷 서비스를 위해 매우 현실적인 방법이라 할 수 있다.

그러나 이러한 기술들의 성능을 향상시키기 위하여 중요한 이슈가 될 수 있는 복제 서버의 배치 전략에 관한 연구는 활발하지 않은 상황이다. 적절한 수의 복제 서버의 할당과 배치는 시스템의 전체적인 성능에 크게 영향을 미칠 수 있으며 사용자가 웹 서버에 접근하는데 드는 비용과 지연 시간을

※ 본 논문을 한국과학재단 특정기초(과제번호 : R01-2001-000-00362-0) 연구비 지원에 의해 수행되었음.
† 정 회 원 : 동덕여자대학교 컴퓨터학과 교수
†† 준 회 원 : 숭실대학교 대학원 컴퓨터학과
††† 종신회원 : 숭실대학교 컴퓨터학과 교수
논문접수 : 2004년 1월 19일, 심사완료 : 2004년 4월 12일

줄일 수 있다. 기존에 캐쉬와 프락시 서버의 위치 선정에 관한 연구들은[5-11] 최적의 프락시 서버 수를 알아내는 greedy 알고리즘과 선택된 프락시 서버를 적절한 곳으로 배치하기 위한 NP-hard 문제로 접근하여 해결하고 있으나 복잡한 계산 시간을 필요로 하며 복제 웹 서버와는 성격이 다르기 때문에 그대로 적용하기는 어려운 상황이다.

이에 본 연구에서는 인터넷상의 노드들이 트리 구조를 이루고 있다고 가정하고 bottom-up 다이나믹 프로그램 접근 방법으로 적절한 복제 서버의 개수와 위치를 찾기 위한 알고리즘을 제안한다. 이러한 연구는 최근 활발히 연구되고 있는 CDI(Content Delivery Internetworking)[4]에서 CDN 네트워크 간의 요청 리다이렉터 배치 문제에도 그대로 적용할 수 있다.

본 연구의 목적은 복제 서버로의 접근 비용을 최소화하고 효율을 높이기 위하여 다음의 문제를 해결하였다.

- 얼마나 많은 수의 복제 서버를 배치해야 하는가?
- 복제 서버를 인터넷상의 어느 위치에 배치할 것인가?

2. 관련 연구

복제 웹 서버의 적절한 수와 위치 선정 문제는 기존의 프락시 서버의 배치 문제와 연관지어 생각해 볼 수 있는데, 대부분의 기존 프락시들은 지연을 줄이기보다는 네트워크 혼잡을 줄이는 것에 초점을 두고 보통 직관적으로 중요한 지점이라고 생각할 수 있는 LAN의 라우터나 인터넷의 게이트웨이 또는 기업이나 조직의 전략적 위치에 배치되었다[13].

그러나 분산 웹 서버 환경에서 효율적으로 사용자의 접속 지연을 줄이고 네트워크 대역폭 사용을 줄이기 위해서는 얼마나 많은 수의 복제 서버를 네트워크의 어디에 위치시킬 것인가에 관한 연구가 필요하다. 복제 서버를 사용자 가까이에 두어 지연을 줄이는 것과 트래픽이 많은 곳에 두어 요청 접속률을 높이는 것 사이에는 역 상관관계(tradeoff)가 있다.

Qiu et al.(2001)은 n개의 노드에서 CDN 서버가 될 수 있는 k개의 노드를 선택하는데 있어서 특정 노드 j가 CDN 서버에 할당되는 경우, d_j를 노드 j에 대한 수요, c_{ij}를 노드 i와 j사이의 거리라고 할 때 비용 d_jc_{ij}를 최소로 하는 k를 선택함으로써 최적의 위치를 알아내는 연구를 하였다. 몇 가지 알고리즘을 제시하여 비교 하였는데 시간 복잡도의 비교는 <표 1>과 같고 AS(Autonomous System) 레벨의 인터넷 토폴로지를 모델링하여 시뮬레이션 한 결과 거리와 요청 부하를 기반으로 greedy 알고리즘에 의한 배치가 가장 좋은 성능을 나타내었다.

〈표 1〉 시간 복잡도 비교

Tree-based [9]	Greedy	Random	Hot Spot
$O(N^3M^2)$	$O(N^2M)$	$O(NM)$	$N^2+\min(NlogN, NM)$

Li et al.(1998, 1999)은 웹 프락시의 배치를 위한 연구를 하여 노드들이 선형 토폴로지를 이룰 때와 트리 토폴로지를 이룰 때 시스템 자원과 트래픽 패턴을 고려하여 지연을 최소화 할 수 있는 알고리즘을 제안하였다. 여기에서는 프락시의 개수가 M개로 지정되어 있음을 가정하고 다이나믹 프로그래밍 알고리즘을 이용하여 M개 프락시의 배치 문제를 해결하였으나 트리 토폴로지의 경우 N개의 노드에 M개의 프락시 서버를 배치하는데 $O(N^3M^2)$의 시간 복잡도를 필요로 한다.

Jamin et al.(2000)과 Radoslavov et al.(2001)는 클라이언트와 복제 웹 서버의 거리를 최소화 할 수 있는 복제 서버의 배치 문제를 *k-median problem*으로 접근하여 해결하였으며 BGP 라우팅 테이블을 이용하여 실제 인터넷 토폴로지를 모델링하여 시뮬레이션 하였다. Radoslavov et al.(2001)는 보다 정확한 결과를 위하여 AS와 라우터 레벨의 인터넷 토폴로지 상에서 모델링하여 degree가 큰 노드에 복제 서버를 배치하는 것이 greedy에 비해서 성능이 더 좋음을 보였다. 그러나 모든 노드가 비슷하게 degree가 적은 경우는 적용하기에 적합하지가 않다.

Jamin et al.(2001)는 AS를 하나의 노드로, AS간의 연결을 링크로 보고 AS 레벨의 토폴로지 상에서 복제 서버의 수가 성능 향상에 얼마나 영향을 주는지를 연구하여 복제 서버의 수가 많을수록 성능에는 효과적임을 보여 주었다. 그러나 AS의 크기는 매우 다양하므로 AS를 단지 하나의 노드로 처리하여 배치라는 것은 적당하지 않다고 볼 수 있다.

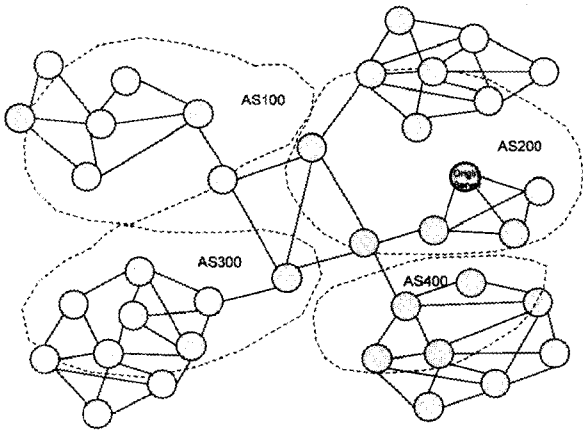
Krishman et al.(2000)는 선형, 링 토폴로지에서의 캐쉬 프락시 배치 알고리즘을 제안하였다. 각 알고리즘의 시간 복잡도는 $O(NM^3)$, $O(NM^4)$로 나타났다.

3. 복제 서버 배치 알고리즘

3.1 네트워크 모델

제안하는 알고리즘은 (그림 1)과 같이 원본 웹 서버를 루트로 하는 트리 구조를 가정하여 트리의 어떤 노드에 복제 서버를 배치 할 것인지를 bottom-up 다이나믹 프로그래밍 방법에 의해 해결하고자 한다. 트리를 서브 트리로 세분화하여 하위 레벨에서부터 트래픽 양과 지연시간에 따른 비용이 정해진 임계값을 초과하지 않도록 복제 서버를 배치함으로써 복제 서버의 적정한 위치와 개수를 도출한다. 결과적으

로 라우터 레벨의 토폴로지를 사용할 경우 각 AS별로 적정 수의 복제 서버가 할당될 것이며 이것은 모든 사용자에게 만족할 만한 서비스를 제공해 줄 수 있다.



(그림 1) 네트워크 모델

제안하는 모델의 하부 구조는 하위 노드에서 상위 노드로 요청을 전달하는 트리 토폴로지이고 모든 클라이언트는 언제나 가장 비용이 적은 복제 서버를 선택함을 가정하며 각 복제 서버가 서비스 할 수 있는 최대 클라이언트 수는 고려하지 않는다. 본 모델에서는 (그림 2)와 같은 표기를 사용하며 노드들의 네트워크를 그래프 G로 나타낸다.

Given a graph $G = (V, E)$,
V : *Set of Nodes*
E : *Set of Links for* $E \subseteq V \times V$
S : *Origin Server*, $S \in V$
N : *Number of Nodes*
R : *Set of Replicas*, $= \{r_1, r_2, \cdots, r_m\}$
$G = T_1 \cup T_2 \cup \cdots T_m$, T_i : *Shortest Path Tree Rooted from node i,*
$1 \le i \le m$, $T_i \cap T_j = \phi$ *for* $i \neq j$
$|R| = |r_1| + |r_2| + \cdots + |r_m| = k$, *k a is Number of Replicas in G*

(그림 2) Notations

3.2 복제 서버 배치 알고리즘

N개의 노드로 이루어진 트리에서 임의의 노드 v는 자신과 하위 노드간의 거리 차이에 의해 생기는 지연 시간과 하위 노드로부터 발생되어 지나가는 트래픽이 발생하게 된다. 그러므로 $v \in V$에 대해 각 노드의 비용은 해당 노드를 지나는 트래픽의 양(ω)과 하위 노드와의 거리 차이에서 생기는 링크의 시간 지연(d)으로 나타낼 수 있다.

링크의 지연시간은 패킷이 해당 링크를 경유하는데 소요되는 시간이며 전송지연, 전파지연의 합이다. 그러므로 트리 구조에 있어서 노드 v의 비용은 식 (1)과 같이 노드 v의 자식 노드들에서 발생되는 트래픽 양과 지연 시간의 곱과 자식 노드들의 비용의 합으로 나타낼 수 있다.

$$Cost(v) = \sum_{i \in Ch_v} (\omega(i) * d(i, v) + Cost(i)) \quad (1)$$

Ch_v는 v의 자식 노드들을 의미하며 v의 비용이 임의로 정한 서비스 임계값($\mathcal{L}$) 보다 크면 자식 노드들 중 비용이 가장 큰 노드가 복제 서버가 된다. 그러면 선택된 복제 서버 노드를 루트로 하는 트리는 노드 집합에서 제외시키고, 비용이 임계값 보다 작은 경우는 해당 트리에서는 복제 서버가 필요 없는 것으로 보고 상위 레벨로 올라간다. 다시 상위 레벨의 노드에서 비용을 구하여 원본 서버를 만날 때까지 반복한다.

임계 값은 (그림 3)과 같이 지연과 대역폭의 곱으로 정할 수 있는데 지연값은 이전의 지연 평균과 현재의 지연을 고려하며 대역폭은 현재 사용가능한 대역폭을 의미한다. 임계값은 다음과 같이 결정된다.

임계값($\mathcal{L}$) = $d_{avg'} * \overline{\omega}$

$\overline{\omega} = B_{max} * \alpha$, $0 < \alpha < 1$
B_{max} : 해당 링크에서 허용하는 최대 대역폭
α : 현재 허용하는 대역폭 사용 비율

davg' : $\beta * d_{avg} + (1-\beta) * d_{new}$, $0 < \beta < 1$
d_{avg} : 종전까지 연결된 노드들과의 평균 지연값
d_{new} : 현재의 평균 지연값

(그림 3) 임계값 결정 식

임계 값을 설정함으로써 모든 노드가 최대 대역폭을 넘지 않으며 지연 값에 대해서는 지수함수를 사용함으로써 일정 수준의 지연을 보장할 수 있다. bottom-up 다이나믹 프로그램 접근 방법에 의한 문제 해결 알고리즘은 (그림 4)와 같다.

$\mathcal{L}$: Threshold
$\omega(i)$: Traffic amount of node i
$d(i, v)$: Delay between node i and v
Ch_v : Set of child nodes of node v
l : Depth of tree
k : Number of replicas

for each node v at level l
  if $l = 0$ stop ;
  $Cost(v) = \sum_{i \in Ch_v} (\omega(i) * d(v, i) + Cost(i))$
  if $Cost(v) > \mathcal{L}$
    $\overline{v} = \max[Cost(i)]_{i \in Ch_v}$;
    $\gamma_k = \overline{v}$;
    $V = V - (\{\gamma_k\} \cup Ch_{\gamma k})$;
    k++ ;
    endif
l -- ;

(그림 4) 제안하는 알고리즘

여기서 k는 복제 서버의 개수가 되며 $k=1$의 의미는 해당 트리에 복제 서버가 존재하지 않고 원본 서버만 존재하는 것이다. 만약 $k=N$이 되면 모든 노드가 복제 서버가 되어야 하는 것이다. 위 알고리즘을 통해 필요한 최소의 복제 서버 개수 k와 최적의 복제 서버 위치를 알 수 있게 된다.

[정리 1] 지연시간, 트래픽 양, 임계 값이 주어졌을 때 제안한 알고리즘에 의해 구해진 복제서버의 개수 k는 임계 값 $\mathcal{L}$을 만족하기 위한 최소값이다.

[증명] 만약 최소 개수가 k가 아니라면 k보다 작은 복제 서버의 수로도 서비스를 만족할 수 있다는 것이다. 최소 개수가 k-1개가 되기 위해서는 임의의 복제 서버 r_i와 r_j중에 하나가 복제 서버 집합에서 제외되어야 한다. 그렇게 되면 i 또는 j 서버의 부모 노드의 비용이 임계 값 $\mathcal{L}$를 초과하게 되어 서비스를 만족할 수 없다.

[정리 2] 서브 트리 Ti의 모든 노드 v에서 가장 가까운 복제 서버 r_k에 접근하는데 드는 비용은 임계 값 $\mathcal{L}$를 초과하지 않는다.

[증명] 만약 임계 값 $\mathcal{L}$를 초과하게 되면 v에서 복제 서버 r_i에 이르기 전에 다른 복제 서버가 존재해야 한다. 그러나 그렇게 되면 복제 서버의 수는 최적의 개수인 k 보다 1개 더 많은 $k+1$이 되므로 [정리 1]에 위배된다. 그러므로 서브 트리 Ti의 모든 노드 v에서 가장 가까운 복제 서버 r_i에 접근하는데 드는 비용은 임계 값 $\mathcal{L}$를 초과하지 않는다.

4. 성능 분석

본 논문에서는 복제 서버의 적절한 개수 및 위치 선정을 위한 알고리즘을 제안하였다. 성능 분석을 위하여 먼저 복제 서버로의 접근 비용과 임계값을 고려하여 복제 서버를 배치하는 제안하는 방법과 임의로 복제 서버를 배치하는 방법의 효율성을 수 예제를 통해 비교하고, 시뮬레이션으로 검증하였다. 링크의 지연 값은 실제 실험을 통해 얻어진 평균 지연 값을 평균으로 하는 지수분포에 따라 전송 지연 값을 임의로 생성하였다. 각 링크상에 발생되는 트래픽 값 또한 임의의 확률로 생성하였다. 그리고 제안한 알고리즘을 수행하기 위해 필요한 시간 복잡도를 도출하여 기존에 제안된 알고리즘과 비교하였다.

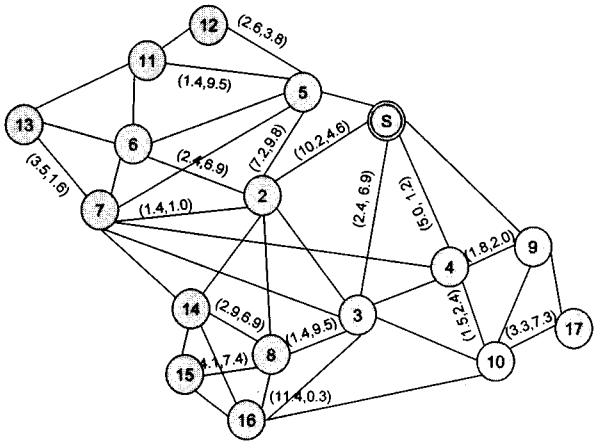
4.1 예제를 통한 분석

(그림 5)와 같은 네트워크 모델에서 제안한 방법에 의해 복제 서버를 배치한다. 제안하는 모델은 트리를 형성하고 루트(S)는 원본 웹 서버를 의미한다. 접근 비용은 각 호스트들이 원본 서버를 포함한 각 복제 서버까지 접근하는데 드는 비용을 의미하며 각 링크 상의 지연값은 실제 ping 테스트로 100여개 대학 사이트를 상대로 측정된 평균 지연 값 9.02 [12]을 $\frac{1}{\lambda}$로 하는 지수함수 분포에 의해 임의로 생성한다. 지연값을 생성하는 식은 다음 식 (2)와 같다.

$$d=-\frac{\ln R}{\lambda},\ 0<R<1 \tag{2}$$

(그림 3)의 임계값 결정 식에 의해 지연값 평균이 3.9, 최대 트래픽 양이 9.8이고 α를 0.8, β를 0.2라고 할때, 임계값 $\mathcal{L}$는 24.5가 된다.

4.1.1 제안한 방법에 의한 서버배치



(그림 5) 예제 네트워크 모델

(그림 5)의 예제를 통해 복제 서버의 개수와 위치를 알아본다.

( )안의 값은 노드 i의 트래픽 양과 i의 부모 노드인 v와의 지연시간을 나타낸다. 즉, ($\omega(i)$, $d(i, v)$) 값이다. 임계값은 24.5로 하고, 잎 노드를 제외한 트리의 최하위 레벨의 노드부터 비용을 구하여 복제 서버의 위치를 선정하면서 최상위 노드인 원본 웹 서버 S까지 올라간다.

1단계 : 5, 6, 7, 8, 9, 10 노드의 비용을 구한다.

$Cost(5) = 1.4\times9.5 + 2.6\times3.8 = 23.2$
$Cost(6) = 0$
$Cost(7) = 3.5\times1.6 = 5.6$
$Cost(8) = 2.9\times6.9 + 4.1\times7.4 + 11.4\times0.3 = 53.8$
$Cost(9) = 0$
$Cost(10) = 3.3\times7.3 = 24.1$

노드 8의 비용이 임계값 24.5를 초과했으므로 8의 자식 노드 중 트래픽과 지연시간이 가장 큰 15번 노드가 복제 서버 r_j이 되고 15번 노드는 트리에서 제외된다.

2단계 : 트리의 한 레벨 위로 올라가서 2, 3, 4 노드의 비용을 구한다.

$Cost(2) = (7.2\times9.8+1.4\times9.5+2.6\times3.8)+(2.4\times6.9)+$
$(1.4\times1.0+3.5\times1.6) = 117.3$
$Cost(3) = (1.4\times9.5) + (2.6\times6.9 + 11.4\times0.3) = 34.7$
$Cost(4) = (1.8\times2.0) + (1.5\times2.4 + 3.3\times7.3) = 31.3$

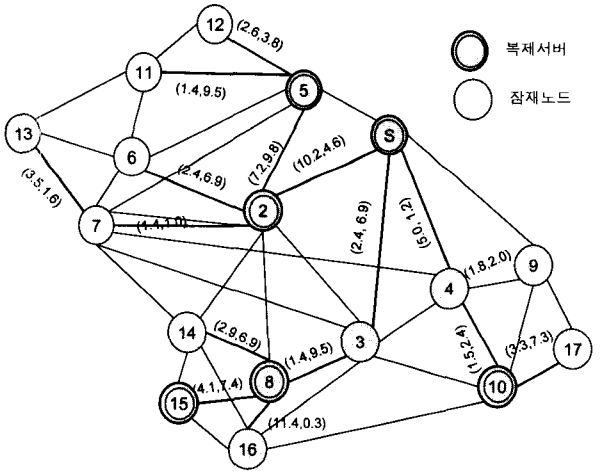
2, 3, 4번 노드 모두가 임계값을 초과하므로 각 자식 노드들 중 비용이 가장 큰 5, 8, 10번 노드가 복제 서버가 되고, 해당 노드를 루트로 하는 서브 트리는 전체 트리에서 제외된다.

3단계 : 다시 한 레벨 위로 올라가서 원본 서버인 노드 S의 비용을 구한다.

$Cost(S) = (10.2\times4.6+2.4\times6.9+1.4\times1.0+3.5\times1.6)+$
$(2.4\times6.9)+(5.0\times1.2+1.8\times2.0) = 96.6$

S의 비용이 임계값을 초과하므로 자식 노드들 중 비용이 가장 큰 2번 노드가 복제 서버로 선택된다.

그래서 (그림 6)과 같이 복제 서버의 위치는 2, 5, 8, 10, 15번 노드가 되며 복제 서버의 개수 k는 5가 된다.



(그림 6) 선택된 복제 서버

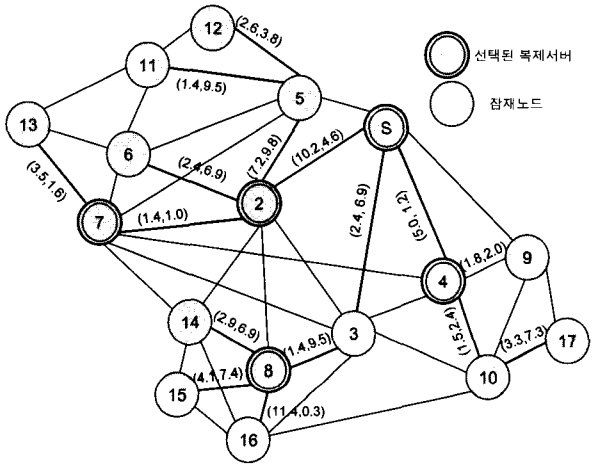
4.1.2 임의의 복제 서버 선택

(그림 7)과 같이 임의로 난수표를 이용하여 2, 4, 7, 8 노드를 복제 서버로 선택한 경우 각 복제 서버의 접근 비용은 다음과 같다.

$Cost(2) = (7.2\times9.8 + 1.4\times9.5 + 2.6\times3.8) + (2.4\times6.9)+$
$(1.4\times1.0 + 3.5\times1.6) = 117.3$
$Cost(4) = (1.8\times2.0) + (1.5\times2.4 + 3.3\times7.3) = 31.3$
$Cost(7) = 3.5\times1.6 = 5.6$
$Cost(8) = 2.9\times6.9 + 4.1\times7.4 + 11.4\times0.3 = 53.8$

이 경우 비용을 고려하지 않고 복제 서버를 선택함으로써 복제 서버 2, 4, 8의 비용은 임계값을 초과하고 있으며 반면, 복제 서버 7의 경우는 다른 복제 서버에 비해 서비스 부담이 매우 적어 효율성이 떨어진다. 이런 환경은 모든 클라이언트에게 고르게 안정적인 서비스를 제공할 수가 없다.

제안한 알고리즘에서는 각 복제 서버의 비용 효율이 적정한계를 초과하지 않으며 일정한 지연값과 대역폭을 만족하는 복제 서버로 클라이언트가 접근할 수 있도록 복제 서버를 배치함으로써 최소의 복제 서버의 개수로 안정적인 서비스가 가능하다. 반면 임의로 복제 서버를 선택하는 경우는 복제 서버의 비용 효율의 차이가 극심할 수 있으며 이는 복제 서버의 효율을 떨어뜨리고 클라이언트에게 안정적인 서비스를 제공할 수가 없음을 의미한다.



(그림 7) 임의로 선택한 복제 서버

4.2 시뮬레이션에 의한 결과 비교

시뮬레이션에 사용할 모델은 (그림 5)를 그대로 사용하고, 매 실험마다 지연과 대역폭 값을 생성하여 임계 값과 모든 노드의 초기 비용을 구하고 제안한 알고리즘에 의해 임계 값을 초과하지 않도록 복제 서버를 배치한 후의 각 노드의 비용을 구하여 노드 비용의 최소, 최대, 평균값을 비교 분석 하였다.

실험 결과 제안한 알고리즘을 적용하기 전에는 비용의 변이 계수 평균이 187%인데 반해 제안한 알고리즘을 적용한 경우는 변이 계수 평균은 91%로 비용 편차가 적은 것으로 나타났다. <표 2>에서 볼 수 있는 것과 같이 제안하는 알고리즘을 적용하기 전에는 각 노드의 비용 편차가 매우 크며 당연히 원본 서버 S의 비용은 가중되는 것을 알 수 있다. 그러므로 그런 상태에서 임의로 복제 서버를 선택하게 되는 경우는 최선의 선택을 할 수가 없다. 그러나 제안한 방법에 의해 복제 서버를 배치한 경우 각 노드의 비용은 임계 값 안에서 거의 편차 없이 일정하며 <표 3>과

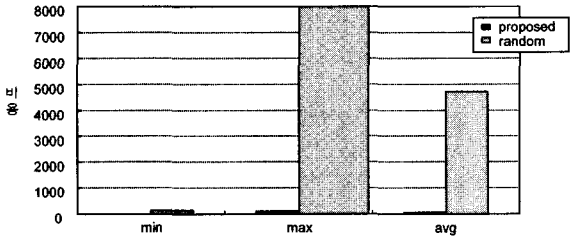
〈표 2〉 각 노드의 비용(알고리즘 적용 전) $\alpha=0.8$, $\beta=0.2$

횟 수	delay	band width	잎 노드를 제외한 각 노드의 비용							
			S	2	3	4	5	7	8	10
1	8.97	12.23	1165.15	355.79	123.84	99.11	27.82	1.24	67.48	18.15
2	7.07	16.15	2769.21	634.35	324.98	617.63	101.25	19.78	120.73	81.30
3	6.61	12.63	5427.07	926.32	508.0	1194.37	231.45	39.33	168.99	100.78
4	4.26	12.21	9023.16	1206.12	778.31	1567.23	233.73	39.37	264.70	111.21
5	8.16	12.30	14212.59	1924.91	1415.45	1682.99	480.56	59.10	602.63	113.91
6	6.02	17.20	21317.73	3041.23	2127.86	1880.76	548.0	60.27	618.74	188.56
7	10.63	24.75	31856.74	4121.10	4022.93	2098.08	606.87	63.40	873.42	189.65
8	9.43	14.62	45243.26	5412.80	5295.29	2512.52	832.39	361.64	1270.10	199.41
9	10.77	33.62	64001.36	7221.91	6688.66	2844.55	842.52	361.66	1385.20	301.69
10	9.36	23.29	85344.10	8704.72	8279.83	4300.65	988.94	445.74	1561.15	551.68

같이 평균 복제 서버의 수(k)는 4가 되며 각 노드의 최대, 최소 비용의 차이나 평균 노드 비용이 알고리즘을 적용하기 이전에 비해 월등히 적으며 그 값도 일정하게 작다는 것을 알 수 있다. 알고리즘을 적용하지 않을 경우 노드 비용 최대값의 평균은 28036.04로 다른 값에 비해 상당히 커서 (그림 8)의 그래프에 모두 표시하지 못하였다.

〈표 3〉 알고리즘 적용 전, 후의 각 노드 비용 비교
$\alpha=0.8$, $\beta=0.2$

횟수	$\mathcal{L}$	제안한 알고리즘 적용				알고리즘 적용 전		
		k	min	max	avg.	min	max	avg.
1	70.27	7	0	67.48	17.97	1.24	1165.15	232.32
2	96.33	4	0	82.52	37.78	19.78	2769.21	583.65
3	67.80	4	0	62.98	35.05	39.33	5427.07	1074.54
4	46.23	3	0.04	23.57	10.50	39.37	9023.16	1652.98
5	72.63	6	2.70	70.77	31.55	59.10	14212.59	2561.52
6	88.70	3	0	83.86	41.20	60.27	21317.73	3722.89
7	192.22	4	0	187.63	63.98	63.40	31856.74	5479.02
8	113.13	5	0	101.38	38.59	199.41	45243.26	7640.93
9	282.40	3	0.02	240.91	107.92	301.69	64001.36	10455.94
10	179.68	4	0	175.94	81.88	445.74	85344.10	13772.10
평 균		4	0.28	109.70	46.64	122.93	28036.04	4717.59
편차 평균			41.327			9512.86		
변이계수			0.913			1.873		



(그림 8) 알고리즘 적용 전 후의 비용 비교

4.3 시간 복잡도 비교 및 비용 효율 검증

[정리 3]에서 보듯이, 제안한 알고리즘의 시간 복잡도는 $O(n^2)$보다 적다. 따라서 <표 1>에서 정리된 기존의 제안된 알고리즘과 비교할 때, [7]의 연구와 [5]의 greedy 방법보다 좋은 성능을 나타냄을 알 수 있다.

[정리 3] 제안한 알고리즘의 시간 복잡도는 $O((n-d^{h})\cdot|ch_v|)$이고, 이것은 $O(n^2)$ 보다 작다.

여기서, $|ch_v|=\frac{d^{h-l+1}-1}{d-1}-1$, $d>1$이다.

d : 트리의 차수, h : 전체 트리의 깊이, l : 노드 v의 레벨

[증명] (그림 4)의 알고리즘에 의해 각 노드가 자신의 비용을 계산하기 위하여 자신의 서브 트리 상에 존재하는 자식 노드 리스트의 크기 $|ch_v|$만큼 비용을 검색해서 임계 값 초과여부를 확인하여야 하므로 $O(|ch_v|)$의 시간이 필요로 하며, 원본 웹 서버인 근 노드까지 탐색하게 되므로 $O((n-d^{h})\cdot|ch_v|)$의 시간 복잡도를 갖는다.
또한, $n-d^{h}<n$이고 $|ch_v|<n$이므로 $O((n-d^{h})\cdot|ch_v|)<O(n^2)$은 명백하다.

[정리 4] 각 복제 서버들의 비용 효율(ρ)을 식 (1)과 같이 정의할 때,

$$\rho = \frac{Cost_k}{\sum_{i=1}^{m} Cost_i} \text{ for } \exists k < n, \forall i < n \quad (1)$$

제안한 알고리즘에 의해 배치된 각 복제 서버들의 비용 효율(ρ)은 다음 식 (2)를 만족한다.

$$0 < \rho < \frac{\pounds}{\sum_{i=1}^{m} Cost_i} \quad (2)$$

[증명] 노드의 집합 $V = \{v_1, v_2, \cdots, v_n\}$에 대해 식 (1)에 의해 다음 식 (3)을 만족한다.

$$\text{for, } \exists v_k \in V, \ \rho = \frac{Cost_{v_k}}{\sum_{i=1}^{m} Cost_i} (0 < m < n) \quad (3)$$

제안한 알고리즘에 의해 $Cost_{v_k} \leq \pounds$, $Cost_{v_k} > 0, \pounds > 0$,이고 $Cost_i > 0$이다. 그러므로 다음 식 (4)가 만족되고 [정리 4]가 참임이 증명된다.

$$\rho = \frac{Cost_{v_k}}{\sum_{i=1}^{m} Cost_i} \leq \frac{\pounds}{\sum_{i=1}^{m} Cost_i} \quad (4)$$

5. 결론 및 향후 연구 계획

본 연구에서는 적정한 복제 서버의 수와 위치 선정에 관한 문제를 다루었다. 문제의 해결 방법은 트리 토폴로지 상에서 다이나믹 프로그래밍 방법에 의해 각 복제 서버가 서비스하는 비용이 일정 수준을 넘지 않도록 복제 서버를 배치하였다. 연구 결과로 $O((n-d^h)\cdot|ch_0|)$의 시간 복잡도 안에 서버 배치 문제를 해결하였으며 제안한 알고리즘이 지연과 트래픽에 기반한 비용 면에서 안정적인 서비스를 보장함을 증명하였다.

그러나 인터넷 토폴로지가 트리임을 보장할 수 없으므로 향후 BGP 테이블을 이용한 AS 레벨의 실제 인터넷 토폴로지를 이용하여 시뮬레이션하여 검증하는 것이 필요하다.

참 고 문 헌

[1] M. Rabinovich and O. Spatscheck, "Web Caching and Replication," Addison Wesley Professional, 2002.

[2] G. Peng, "CDN : Content Distribution Network," In Tech. Reports, TR-125, School of Computer Science Department, Stony Brook University, 2002.

[3] S. Saroiu, P. K. Gummadi and S. D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," In proceedings of Multimedia Computing and Networking, 2002.

[4] A. Biliris, C. Cranor, F. Douglis, M. Rabinovich, S. Sibal, O. Spatscheck, and W. Sturm, "CDN Brokering," In proceedings of WCW, 2001.

[5] L. Qiu, V. N. Padmanabhan, G. M. Voelker, "On the Placement of Web Server Replicas," In proceedings of IEEE INFOCOM, 2001.

[6] B. Li, X. Dong., M. J. Golin, K. Sohraby, "On the Optimal Placement of Web Proxies in the Internet : Linear Topology," In proceedings of HPN, 1998.

[7] B. Li, M. J. Golin, G. F. Italiano, X. Deng, and K. Sohraby, "On the Optimal Placement of Web Proxies in the Internet," In proceedings of IEEE INFOCOM, 1999.

[8] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, "On the Placement of Internet Instrumentation," In proceedings of IEEE INFOCOM, 2000.

[9] P. Radoslavov, R. Govindan, and D. Estrin, "Topology-Informed Internet Replica Placement," In proceedings of WCW, 2001.

[10] S. Jamin, C. Jin, A. Kurc, D. Raz and Y. Shavitt, "Constrained Mirror Placement on the Internet," In proceedings of IEEE INFOCOM, 2001.

[11] P. Krishnan, D. Raz and Y. Shavitt, "The Cache Location Problem," IEEE/ACM Transactions on Networking, Vol.8, No.2, pp.568-582, 2000.

[12] J. H. Park, "An Improved Overlay Multicast Scheme for Enhancing Transport Efficiency of High Capacity Contents," Soongsil University, Seoul, Korea, Master's thesis, 2003.

[13] M. Nabeshima, "The Japan Cache Project : An Experiment on Domain Cache," In proceedings of the Sixth International WWW Conference, 1997.

김 선 호

e-mail : shkim98@dongduk.ac.kr

1987년 이화여자대학교 사범대학 수학교육 전공 학사

1992년 이화여자대학교 교육대학원 전자계산교육전공 석사

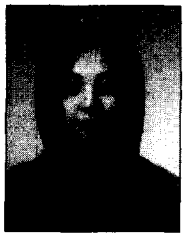
2004년 숭실대학교 컴퓨터학과 박사과정 졸업예정

1987년~1989년 대우전자부품(주) 전산실

1990년~1993년 한국생산성본부 정보화사업부

1998년~현재 동덕여자대학교 정보과학대학 강의전임교수

관심분야 : Internet Protocol, Mobile IP, CDN, DRM



윤 미 연

e-mail : myyoon@cherry.ssu.ac.kr

2000년 가톨릭대학교 수학과/컴퓨터학과 학사

2002년 숭실대학교 컴퓨터학과 공학석사

2002년~현재 숭실대학교 컴퓨터학과 박사과정

관신분야 : 멀티캐스트, 실시간프로토콜, Wireless Communication, 정보보호, CDN

신 용 태

e-mail : shin@comp.ssu.ac.kr

1985년 한양대학교 산업공학과 학사

1990년 Univ. of Iowa 컴퓨터학과 석사

1994년 Univ. of Iowa 컴퓨터학과 박사

1994년~1995년 Michigan State Univ. 전산학과 객원교수

1995년~현재 숭실대학교 컴퓨터학부 부교수

관심분야 : 멀티캐스트, 그룹통신, 인터넷 보안, 이동 인터넷 통신