비구조적인 피어-투-피어 네트워크상에서 효율적인 복제 기법[1)]

최우락 이문수° 박성용
서강대학교 컴퓨터학과
lazylune@dcclab.sogang.ac.kr, lilo@sogang.ac.kr°, parksy@sogang.ac.kr

An Efficient Replication Strategy in Unstructured Peer-to-Peer Networks

Wurak Choi, Moonsoo Lee°, Sungyong Park
Dept. of Computer Science, Sogang University

요 약

초기의 비구조적인 피어-투-피어 방식은 플러딩(flooding) 방식의 검색 기법을 사용하는데, 이 기법은 통신비용이 매우 많이 소요되어 비효율적이다. 따라서 효율적인 검색 기법으로 임의 경로(random walk) 검색 방식이 제안되었다. 이 검색 기법은 메시지가 임의로 노드를 이동하기 때문에, 검색의 성공률이 낮다. 이를 보완하기 위하여 효율적인 복제 기법이 요구된다. 현재 나와 있는 복제 기법은 여러 방법이 있으나, 모두 통신에 고비용을 요구한다. 따라서 복제 기법에서는 통신비용을 최소화하는 효율적인 복제 기법이 필요하다. 본 논문에서는 캐시를 사용하여 직접적인 데이터 통신 비용을 최대로 줄이는 한편, 복제를 질의가 많이 도착하는 곳에 위치시켜 검색률의 저하를 막고, 잘못된 캐시 관리 기법을 통해 동적인 환경에서도 잘못된 캐시로 인한 검색의 실패를 최소할 수 있는 기법을 제안한다.

1. 서 론

피어-투-피어 네트워크는 중앙 집중화(centralized) 방식과 분산화(distributed) 방식으로 나눌 수 있다. 중앙 집중화 방식은 네트워크 내에 존재하는 모든 컨텐츠에 대한 인덱스를 중앙 서버에서 관리한다. 이런 방식은 단일 지점 장애(single point of failure)의 문제가 있다. 이러한 점을 개선한 분산화 방식은 크게 구조적인(structured) 방식과 비구조적인(unstructured) 방식으로 나눌 수 있다.구조적인 방식은 질의(query)에 대한 검색이 빠르고 항상 응답을 보장 할 수 있는 반면에, 구조적인 네트워크를 유지하기 위한 비용이 많이 들고, 사용자가 자주 참여/탈퇴하는 경우에는 적합하지 않다[1]. 한편 비구조적인 방식은 특정 위상을 구성하지 않고, 맹목적인 검색(blind search) 방식을 이용하여 사용자가 원하는 파일을 가진 피어를 검색한다. 비구조적인 방식은 네트워크를 유지하기 위한 비용이 적게 들지만, 질의에 대한 검색이 느리고, 해당파일이 존재하더라도 검색되지 않는 경우가 발생한다. 이를 보완하기 위해 복제 기법이 필요하다.

기존의 복제 방법으로는 한 단계 복제[1], 경로 복제[2][3], LAR(Lightweight Adaptive Replication) 복제 방법[4]이 있다. 한 단계 복제는 바로 이웃하는 노드들에게 자신의 모든 컨텐츠를 복제하는 방법으로, 검색 성공률을 높일 수 있다. 하지만, 많은 이웃을 가진 노드가 쉽게 과부하 상태가 되게 하며, 복제가 지역적인 부분에서 이루어져 검색의 효율성을 높이기 어려운 문제점이 있다. 경로 복제 방법은 검색에 걸린 홉 수만큼 복제본을 생성한다. 즉, 인기가 있는 컨텐츠는 질의 비율이 높아 네트워크 내에 비교적 많이 존재하기 때문에 검색에 적은 홉 수를 필요로 하고, 비인기 컨텐츠는 질의 비율이 낮아 검색에 많은 홉 수를 필요로 하기 때문에, 질의 변화에 적응적으로 복제를 생성하게 된다. 경로 복제 방법은 크기가 큰 네트워크에서 여전히 복제에 많은 전송 비용을 초래한다는 한계점이 있다. LAR 복제 방법은 많은 양의 복제를 캐시를 통해 대신하여, 앞의 두 복제 기법보다 적은 양의 데이터 통신을 필요로 한다. 하지만 잘못된 캐시에 대한 처리를 하지 않아 동적인 네트워크 상황에서 적응하기 힘들다. 또한 캐시의 질의율을 고려하지 않아 복제의 효율성이 떨어진다.

본 논문에서는 이러한 문제를 해결하기 위해서 검색의 정확성과 속도를 높이고 비용을 줄이는 효율적인 복제 기법을 제시한다. 제안된 기법에서는 캐시를 사용하여 직접적인 데이터 통신비용을 최대로 줄이는 한편, 복제를 질의가 많이 도착하는 곳에 위치시켜 검색률의 저하를 막고, 잘못된 캐시 관리 기법을 통해 동적인 환경에서도 잘못된 캐시로 인한 검색의 실패를 최소할 수 있는 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 복제 기법보다 가볍고, 효율적인 알고리즘을 제시한다. 이를 위해 논의할 시스템 모델을 제시한 뒤, 이 논문에서 제시하는 효율적 복제 알고리즘에 대하여 설명한다. 3장에서는 2장에서 논한 기법에 대하여 시뮬레이션을 통한 실험 결과를 제시하고, 결과를 분석한다. 마지막으로 4장에서는 논문의 결론과 향후 연구 방향에 대하여 기술하도록 하겠다.

2. 효율적 복제 기법

2.1 모델

본 논문에서는 네트워크의 토폴로지를 대부분의 비구조적인 피어-투-피어 네트워크가 형성하는 파워-로 토폴로지[5]로 가정하였다. 제안된 알고리즘에서는 각 노드는 다른 노드와 연결(connection)을 맺음으로써 메시지를 전달 할 수 있다. 연결을 맺은 노드를 이웃(neighbor)이라고 한다. 질의에 대한 검색은 이웃 노드를 통해 전달된다.

복제된 데이터는 캐시(cache)와 복제본(replica)의 두 가지로 나뉜다. 캐시는 실제 데이터에 대한 포인터의 역할을 한다. 즉 실제 데이터에 대한 주소, 크기 등을 저장함으로써 질의 메시지가 쉽게 실제 데이터로 찾아갈 수 있도록 해준다. 반면에 복제본은 실제 데이터를 나타낸다. 따라서 사용자의 전송 요청이 있으면 해당 데이터를 전송해 줄 수 있다.

2.2 알고리즘

효율적 복제를 위한 알고리즘은, 복제에 소요되는 비용을 가능한 최소화해야 한다. 또 쓸데없는 복제를 최소화 하고, 복제를 사용자의 질의가 있는 곳에 배치해 효율성을 높여야 한다. 복제 기법은 지역(local)정보만을 이용해 분산 알고리즘의 확장

1) 사사표시는 현재 논의중이며, 관리규정(협약 체결, 국고지원금 교부시 제시 예정)(안)에는

(국문표기) "이 논문은 2006년도 두뇌한국21사업에 의하여 지원되었음."
(영문표기) "This work was supported by the Brain Korea 21 Project in 2006."로 되어있습니다.

성을 저해해서는 안 된다.

이러한 목표를 달성하기 위하여 제안된 복제 알고리즘은 크게 3가지 알고리즘, 즉, 캐싱을 위한 알고리즘, 잘못된 캐시(stale cache)를 처리하기 위한 알고리즘, 복제 분산 알고리즘으로 구성된다.

캐싱을 위한 알고리즘은 크게 캐시 선택 알고리즘과 노드 선택 알고리즘으로 나눌 수 있다. 캐시 선택 알고리즘은 질의 메시지가 생성되었을 때 전파할 캐시를 선택하는 알고리즘이고, 노드 선택 알고리즘은 메시지가 다른 노드로 전달될 때, 지역적인 정보만으로 해당 노드에 캐시를 전파할 것인지 결정한다. 한편 본 알고리즘에서는 전파 비용 을 줄이기 위해 캐시를 이미 생성된 메시지에 피기-배킹(piggy-backing)하여 전파한다.

잘못된 캐시를 처리하기 위한 알고리즘은 자신이 소유하고 있는 캐시가 잘못된 정보를 가지고 있지 않도록 캐시들을 관리한다. 동적인 네트워크 상황에서는 노드들의 출입이 잦기 때문에, 캐시가 존재하지 않은 노드를 가리킬 수 있다. 이런 경우를 잘못된 캐시라고 하는데, 이는 캐시를 통한 질의 메시지의 라우팅 시 검색율을 낮추는 요인이 된다. 따라서 잘못된 캐시를 처리하기 위한 알고리즘을 통해 질의에 대한 검색율을 높일 수 있다.

복제 분산 알고리즘은 인기 있는 로컬 캐시에 대하여 실제 복제본을 생성한다. 이 때 실제 복제본은 캐시를 소유하고 있는 노드의 요청에 의하여 생성되며, 해당 노드의 이웃 노드에 위치하게 된다. 이 방법이 얻는 이점은 다음과 같다. 보통 연결수가 많은 노드일수록 캐시를 많이 소유할 것이고, 질의 메시지도 많이 전달 받음으로써 인기가 많은 캐시가 발생할 확률이 높다. 따라서 캐시에 대한 복제본을 해당 노드가 직접 소유하는 것은 과부하를 유발하기가 쉽다. 하지만 복제본을 해당 노드의 이웃에 위치시키면, 부하를 분산 시키며, 잘못된 캐시를 처리하기 위한 비용도 적게 소요되는 효과가 있다. 또한, 캐시를 소유하고 있는 노드의 요청에 의해 복제본이 생성됨으로써, 지역적인 질의의 인기도에 적응적으로 복제본을 생성할 수 있다.

2.2.1 캐싱을 위한 알고리즘

캐싱을 위한 알고리즘 중 캐시 선택 알고리즘은 다음과 같다. 먼저 과거 t초 동안 컨텐츠들에 대한 요청이 있었던 경우, 해당 컨텐츠들에 한해서 <식 1>과 같은 확률로 캐시를 만들 컨텐츠를 구한다.

$$P(select(i)) = q_i \qquad \text{<식 1>}$$

여기서 q_i는 특정 컨텐츠의 인기도로, $\sum_i q_i = 1$이 성립한다. 이 방법은 캐시의 생성율(creation rate)을 줄일 뿐만 아니라, 최근의 인기도에 따라 캐시를 생성하기 때문에, 사용자의 질의에 적응적으로 대처할 수 있다.

아울러 캐싱할 노드 선택 알고리즘은 검색 동안에 N개의 노드를 차례로 방문하면서 이웃 노드의 수가 많은 노드라 생각되는 곳에 캐싱한다. 이 문제는 인터뷰 문제로 잘 알려져 있다. 단 기존의 문제와 다른 점은 N이 가변적이라는 점이다. 따라서 인터뷰 문제의 해결 방법을 적용하기 위해서는 N을 예측하는 기법이 필요하다. 여기에서는 가중치 평균을 통해 N을 추정하였다. 가중치 평균에 대한 계산은 해당 노드에서 생성한 질의 메시지에 대한 검색이 종료되면 이루어진다. 가중치 평균을 통해 구해진 N을 N_{avg} 라고 하고, 현재 질의 메시지가 검색에 걸린 홉 수를 N_{new} 라고 할 때 N_{avg}는 <식 2>와 같이 새롭게 계산된다.

$$N_{avg} = \alpha N_{avg} + (1-\alpha) N_{new} \ (\alpha < 1) \qquad \text{<식 2>}$$

캐싱할 노드를 선택하기 위해서 메시지가 k 홉 동안 현재 네트워크 상의 노드들을 돌아다니면서, 그들의 이웃한 노드의 수에 대한 수준을 관찰한다. 여기에서 k는 <식 3>과 같이 구할 수 있다.

$$k = \max_{t \geq 1} \left(\frac{1}{t} + \frac{1}{t+1} + \ldots + \frac{1}{N_{avg}} \right) < 1 \qquad \text{<식 3>}$$

생성된 각 메시지는 k 홉 동안 노드들을 돌아다니면서, 메시지는 가장 많은 이웃 노드의 수 $C_{\max}$를 저장한다. k 홉 이후에 도착한 노드의 이웃 노드의 수가 $C_{\max}$보다 크다면 <식 4>의 확률로 노드에 캐시를 저장하게 된다.

$$P(injectcache) = \min\left(\frac{t}{N_{avg}}, 1 \right) \qquad \text{<식 4>}$$

2.2.2 잘못된 캐시를 처리하기 위한 알고리즘

피어-투-피어 네트워크에서는 노드의 입출이 잦기 때문에, 잘못된 캐시를 처리하기 위한 알고리즘이 필요하다. 하지만 캐시의 무결성을 유지하기 위하여, 캐시가 가리키고 있는 복제본을 소유한 노드를 계속적으로 모니터링 하는 것은 매우 큰 비용을 요구한다. 따라서 확장성 있는 알고리즘을 위해서는 지역적인 정보를 이용하여 잘못된 캐시를 처리하여야 한다. 이를 위하여 여기에서는 캐시에 대한 생존 시간을 설정하는 방법을 사용한다. 즉, 캐시 정보가 캐싱되면 해당 캐시 i에는 생존 시간 $T_{life}(i)$ 가 설정된다. 생존 시간은 시간이 지남에 따라 감소되며, 최종적으로 0 이 되면 해당 캐시는 캐시 테이블에서 삭제된다. 하지만 캐시가 가리키고 있는 컨텐츠가 바로 이웃에 존재하여 네트워크 내에서 활동 중이거나, 혹은 해당 컨텐츠를 소유한 노드로부터 메시지가 전달되어 왔다면, 캐시의 생존 시간을 증가시킨다. 그리고 자신의 이웃 중 한 노드가 네트워크를 떠나게 되면, 캐시 테이블로부터 해당 노드와 관련된 모든 캐시들을 삭제한다.

2.2.3 복제 분산 알고리즘

앞에서 언급한 바와 같이, 이웃 노드의 수가 많은 노드에 컨텐츠를 복사하는 것은 해당 노드를 과부하로 만들기가 쉽다. 따라서 본 알고리즘에서는 복제본을 이웃 노드의 수가 많은 노드의 이웃들 중에 하나를 선택하여 생성한다. q_i가 이웃 i에서 요청이 전달된 비율이고, s_i가 이웃 i의 현재 용량이라고 할 때, i에 복제본이 생성될 확률은 <식 5>와 같다.

$$P(rep_i) = \frac{q_i}{q_{total}} \frac{\frac{s_{total}}{s_i}}{\sum_j \frac{s_{total}}{s_i}} \qquad \text{<식 5>}$$

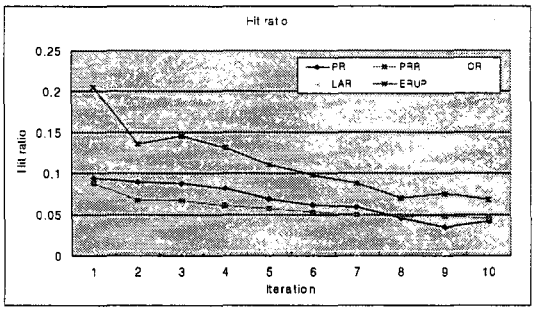
생성 확률을 보면, 요청 비율에 비례하고 현재 용량에 반비례함을 알 수 있다. 즉, 특정 노드에서 해당 요청이 많이 전달된다면, 그 노드에 복제본이 생성될 확률이 높다. 또한 특정 노드가 많은 양의 데이터를 가지고 있다면 해당 노드에게 복제본을 생성하는 것은 피해야 할 것이다.

3. 성능 평가

성능 평가를 위해 비구조적 피어-투-피어 시뮬레이터로 잘 알려진 Neurogrid 시뮬레이터를 확장 개발하여, 논문에서 제시한 효율적 복제 기법과, 앞서 소개한 복제 기법인 한 단계 복제 기법(One hop Replication, 이하 OR), 경로 복제 기법(Path Replication, 이하 PR), 경로 임의 복제 기법(Path Random Replication, 이하 PRR), LAR 등을 직접 모델링하였고, 아무런 복제 기법을 쓰지 않은 임의 경로

검색(Random Walk, 이하 RW)도 모델링하여 각각을 비교 분석하였다.

복제 적중률을 보여주는 [그림 1]은 얼마나 복제본과 캐시의 생성을 최소화 하고, 질의 메시지가 잘 도달하는 곳에 위치시켰는가 하는 값이다. [그림 1]에서 볼 수 있는 것처럼 본 논문에서 제안하는 알고리즘의 복제 적중률이 가장 높게 나타났다. 이는 복제본과 캐시의 생성 수를 최소화하고, 질의 메시지가 잘 도달하는 곳에 위치시킴으로써 나타난 결과라고 할 수 있다. 반면에 많은 수의 복제본을 생성하는 한 단계 복제 방법과 메시지를 생성 시킬 때마다 캐시를 피기-배킹하여 전파하는 LAR 알고리즘은 복제 적중률 면에서 나쁜 성능을 나타냈다.



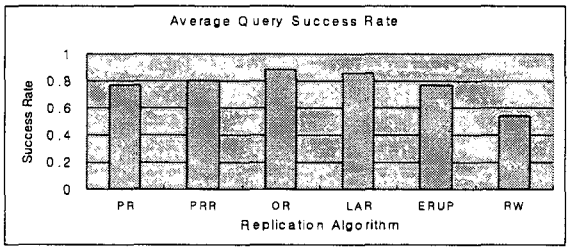
[그림 1] 복제 알고리즘들의 복제 적중률

[표 1]은 복제에 사용된 소요 비용을 측정한 결과이다. 소요비용은 복제본의 생성 개수와 캐시의 생성 개수를 측정한 값이다.

[표 1] 복제 소요 비용

알고리즘	복제본 생성 수	캐시 생성 수
PR	7170	0
PRR	7482	0
OR	108554	0
LAR	1135	267912
ERUP	95	3528

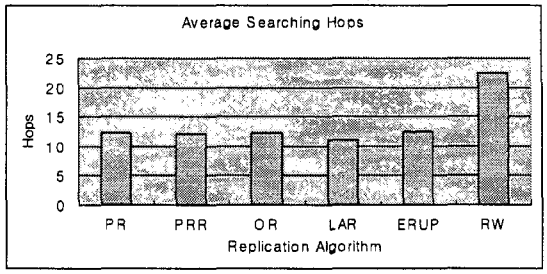
측정 결과 한 단계 복제 알고리즘이 가장 많은 수의 복제본을 생성하여 가장 소요 비용이 컸음을 알 수 있다. LAR은 생성 복제본 수는 적으나 캐시 생성 개수가 많았고, 본 논문에서 제시한 알고리즘은 가장 적은 수의 복제본을 생성하였으며, 캐시의 개수 또한 LAR보다 많은 수를 줄였다.



[그림 2] 복제 알고리즘들의 평균 검색 성공률

[그림 2]의 평균 검색 성공률을 살펴보면, 가장 많은 복제본을 생성하는 OR 알고리즘이 가장 좋은 검색 성공률을 보였다. 본 논문에서 제안하는 알고리즘은 이에 비해 약 13%정도의 성능저하를 보였다. 하지만 그 외의 다른 알고리즘과는 비슷한 검색 성공률을 보임으로써, 복제의 비용을 줄임으로 인한 손실을 최소화하였음을 알 수 있다.

각 알고리즘 별로 평균 검색 거리를 측정한 결과는 [그림 3]과 같다. 아무런 복제 알고리즘을 사용하지 않는 RW 알고리즘의 경우 매우 높은 검색 거리를 보였다. 하지만 나머지 알고리즘들은 비슷한 성능을 보였다. LAR과 본 논문에서 제시한 효율적 복제 알고리즘은 처음에 나쁜 성능을 보이다가 진행 되면서 성능이 향상되어 결국에 다른 알고리즘과 비슷한 성능을 보이게 되었다. 이는 캐시로 인한 효과로 캐시가 전파됨에 따라 검색 거리가 짧아졌음을 알 수 있다. 따라서 복제 소요 비용이 가장 낮았음에도 본 알고리즘이 다른 알고리즘들에 비해 성능이 나빠지지 않았음을 알 수 있다.



[그림 3] 복제 알고리즘들의 평균 검색 거리

4. 결론 및 향후 과제

본 논문에서는 비구조적인 피어-투-피어 시스템의 토폴로지로 잘 알려져 있는 파워-로 토폴로지에 알맞은 효율적인 복제 알고리즘을 제시하였다. 실험 결과 검색 성공률과 평균 검색 거리는 기존의 알고리즘들과 비슷한 성능을 보임과 동시에, 소요 비용 측정 결과 복제에 가장 적은 비용을 소요함을 알 수 있었다. 하지만 본 논문에서는 파워-로 토폴로지를 가정하고, 허브 상에 캐시들을 위치시켰기 때문에, 허브의 잦은 출입은 성능 상에 큰 영향을 미친다. 따라서 추후 노드들의 출입이 잦은 동적인 환경에서 허브의 출입으로 인한 성능 저하를 완화하기 위한 연구가 필요하다.

참고 문헌

[1] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham and Scott Shenker. "Making Gnutella-like P2P Systems Scalable". Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, pp.407-418, August 2003.

[2] Edith Cohen and Scott Shenker. "Replication Strategies in Unstructured Peer-to-Peer Networks". Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications, Volume 32, Issue 4, pp 177-190. August 2002.

[3] Qin Lv, Pei Cao, Edith Cohen, Kai Li and Scott Shenker. "Search and Replication in Unstructured Peer-to-Peer Networks". Proceedings of the International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS 2002, pp 258-259, June 2002.

[4] Vijay Gopalakrishnan, Bujor Silaghi, Bobby Bhattacharjee and Pete Keleher. "Adaptive Replication in Peer-to-Peer Systems". Proceedings of the 24th International Conference on Distributed Computing Systems, pp.360-369, March 2004.

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