P2P 시스템에서의 다중 노드 전역 디렉토리 기반 색인 순차 검색 기법

강인성° 최성진 이화민† 백맹순 황종선
고려대학교 컴퓨터학과 분산시스템 연구실, 고려대학교 사범대학교 컴퓨터교육과†
{iskang°, lotieye, msbak, hwang}@disys.korea.ac.kr, zelkova†@comedu.korea.ac.kr

Indexed-Sequential Search Approach on Multi-Node Global Directory in Peer-to-Peer Systems

InSung Kang, SungJin Choi, MaengSoon Baik, ChongSun Hwang
Distributed Systems Lab., Dept. of Computer Science and Engineering, Korea University
HwaMin Lee†
Dept. of Computer Education, Korea University

요 약

저렴한 비용으로 기존의 클라이언트 서버 시스템을 대체할 수 있는 P2P 시스템에서 서로 공유하고자 하는 파일의 리스트를 얼마나 빠르고 효과적으로 검색하느냐는 시스템 성능을 좌우하는 중요한 항목 중 하나이다. 그러나 기존의 P2P 시스템에서의 검색 기법들은 목적 파일들에 대한 검색 시간의 단축과 키워드 검색이라는 두 가지 설계 목표 중 어느 한 쪽에만 치중하여 설계됨으로써 종합적인 검색 기능이 요구되는 실제 시스템에 적용되기에는 매우 불완전하다. 본 논문에서는 P2P 시스템에 참여하여 파일을 공유하고자 하는 노드들의 모든 공유파일 목록을 하나의 전역 디렉토리로 구성하고 이를 이진 값의 순서대로 저장함으로써 높은 검색 속도로 키워드 검색 기능을 제공하는 다중 노드 전역 디렉토리 기반 색인 순차 검색 기법을 제안한다. 제안된 기법은 높은 검색 속도와 키워드 검색을 지원함으로써 P2P 시스템의 검색 속도 성능 개선과 사용자 편의성을 제공한다.

1. 서론

P2P 시스템은 지역적으로 널리 분산된 많은 노드들을 이용하여 기존의 고 비용의 클라이언트 서버 방식의 시스템을 대체할 수 있는 저 비용 고 효율의 차세대 시스템이다. 이러한 시스템에서 분산된 노드들이 공유하는 파일들에 대한 효율적인 검색은 전체 성능을 가름하는 매우 중요한 사안이다[5].

그러나 기존의 시스템들은, 구조적 P2P 시스템인 Chord[1]처럼 검색 속도가 빠르면 키워드 검색이 불가능하고, Gnutella[7]처럼 키워드 검색이 가능하면 응답 대기 시간이 길어지는 단점을 가진다[5]. 즉, 키워드 검색과 검색속도가 서로 모순(trade off) 관계에 있어 기존의 기법들로 이 두 가지 목표를 모두 달성하는 것은 불가능하다.

본 논문에서는 검색 속도에 있어서 Chord[1], CAN[2] 및 Pastry[3] 등과 같은 구조적인 P2P 시스템과 동등한 성능을 가지며 동시에 키워드 검색을 지원하는 다중 노드 전역 디렉토리 기반 색인 순차 검색 기법을 제안한다.

제안된 기법에서는 모든 공유 파일 목록을 이진 값의 순서대로 정렬하여 이를 여러 개의 디렉토리 노드에 분산 저장한다. 여러 개의 서버를 사용하지만 서버 팜(farm)[8]처럼 한 곳에 집중하여 운영하지 않는다. 각각의 서버가 익명성을 갖고 참여하는 분산된 노드들로 구성되며 노드 1 개당 최고 1024개의 키 값 만을 가지므로 많은 수의 노드가 디렉토리 노드로 참여하게 되어 분산이 원활하게 이루어 진다. 이렇게 구성된 전역 디렉토리는 MS/DOS의 dir 명령과 같이 단순하지만 강력한 검색 기능을 갖는다.

제안된 기법은 한두 개의 서버에서 키워드 검색을 완료하므로 결과를 얻기 위해 많은 노드들로부터 응답을 기다려야 하는 기존의 시스템에 비하여 네트웍의 대역폭을 적게 사용할 뿐 아니라, 구조적인 P2P 시스템과 동등한 속도로 키워드 검색을 함으로써 사용자 편의성을 제공한다.

2. 관련 연구

P2P 시스템에서의 검색 기법에 대한 기존 연구는 크게 분산 해쉬 테이블(Distributed Hash Table)을 사용하는 구조적인 P2P와 비구조적인 P2P로 나뉘는데, 이 두 가지는 서로 메커니즘이나 활용 분야가 상호 보완적이다[4]. 즉 DHT를 사용하는 전자의 경우 검색이 매우 빠르지만 키워드 검색이 불가능하고, 후자는 키워드 검색이 가능하나 검색 효율이 떨어진다.

DHT를 사용하는 Chord[1], CAN[2] 및 Pastry[3] 등의 구조적인 검색 기법은 검색 스텝 수가 적고, 키 값도 전체 노드에 골고루 분산되는 방식이다. 그러나 이러한 방식은 한 번에 단 한 개의 키만을 검색해내므로 키워드 검색이 불가능하다. 뿐만 아니라 하나의 키 데이터를 저장하기 위해서는 각 식별 영역에 최소한 한 개의 노드가 확보되어야 하므로 소규모의 네트웍에서는 운영이 어렵다. 제안된 기법은 키의 개수에 비례하여 디렉토리 노드의 수가 증가한다.

Napster[6]와 같은 비 구조적인 P2P 검색기법에서는 중앙 서버를 두어 사용자들의 공유 파일 목록을 제공하였다. Napster 서버의 폐쇄 조치에 이어 출현한 Gnutella[4][7]는 별도의 서버를 쓰지 않는 순수한 P2P 시스템이지만 검색 질의가 연못에서 퍼져나가는 파동처럼 확산되며 가장자리까지 도달하는 동안 쉽게 대역폭을 소진하므로 확장성이나 응답 속도가 좋지 않다[5]. GIA[4][5]는 동적 토폴로지 적응 등의 메커니즘을 채택하여 Gnutella의 단점을 보완하였으나, 확장성 및 응답속도에 있어 모두 다 DHT 방식에 뒤떨어진다.

3. 전역 디렉토리의 구성

3.1 전역디렉토리의 구조

본 논문에서 제안하는 기법에서는 모든 노드로 MS/DOS의 디렉토리처럼 트리를 구성하는데 루트 노드를 제외한 노드들은

디렉토리의 폴더와 같은 역할을 한다. 이들을 디렉토리 노드라 한다. 이 디렉토리 노드에 그들이 책임을 져야 할 데이터의 리스트를 정해진 개수만큼 키 값에 따라 정렬하여 저장한다. 저장 공간이 다 차면 다른 디렉토리 노드에 나누어 저장을 하고, 각 노드들이 갖고 있는 리스트들을 연결 리스트(linked list) 형태로 연결한다.

루트 노드에는 데이터를 저장하지 않으므로 디렉토리 노드가 아닌 노드들은 모두 동등하게 루트 노드의 역할을 한다.

이 시스템에 처음 참여하는 노드는 자신의 공유 폴더에 있는 파일들의 이름, 크기, 내용의 해쉬 값, 대역폭 및 IP 주소 등의 메타 데이터를 해당 디렉토리 노드에 등록한다. 이 데이터는 파일 별로 해당 디렉토리 노드의 디렉토리 배열(array)에 저장하되, 파일 이름 순서대로 저장한다. 각 노드의 배열은 1 차원이고 1024 개의 방을 갖는다.

저장 위치 검색을 위한 비교 시 모든 대문자는 소문자로 바꾼다. 만일 자료 제공자가 '[ebook]'처럼 키워드의 주위를 []나 <> 등으로 감싸는 경우, 이러한 괄호 글자들도 벗겨낸다. 또 키가 'the'나 'a'와 같은 관사로 시작되는 경우, 이러한 글자들은 잘라내고 비교한다. 그러나 저장할 때에는 원래의 값을 저장한다.

키의 크기는 60자로 제한하여 이를 초과하는 부분은 삭제된다. 60자까지 이름이 같은 파일이라면 같은 내용을 갖고 있을 확률이 매우 높기 때문이다. 이름이 같은 파일들의 상이 여부는 파일의 크기 및 내용의 해쉬 값을 비교함으로써 확인한다.

같은 값을 갖는 키들은 저장 시 해당 방에 IP 주소만을 추가한다. 추가하는 IP 주소의 수는 편의 상 256개까지만 허용한다. IP 주소 256개는 1K의 크기를 차지한다.

3.2 인덱스 테이블

노드 #	0	1	2	..	126	127
인덱스	Abba	Andy	Beatles		Venture	Nill

그림 1. 마스터 인덱스(MI). 트리에서 깊이가 1인 노드들의 인덱스 집합. Nill은 해당 노드가 없음을 뜻한다. 데이터의 양이 적으면 뒤쪽 노드는 존재하지 않는다.

해당 디렉토리 노드에 접근하기 위해서는 그림 1과 같은 인덱스 테이블을 구해야 되는데, 인덱스란 각 디렉토리 노드가 관리하는 배열의 0번 방의 값이며, 인덱스 테이블은 디렉토리 트리에서 깊이가 같은 노드들의 인덱스의 집합이다. 여기서 트리의 깊이가 1인 노드들의 인덱스 집합을 마스터 인덱스(이하 MI)라 한다. 그림 1을 보면 'Beach'라는 값을 갖는 키는 1번 노드의 'Andy'와 2번 노드의 'Beatles' 사이에 있으므로 1번 노드에 있어야 한다.

N #	Cell 0	C1	C2	C3		C1023	Link
0	Abba	able	Ace				L1
1	Andy	art	Baby				L2
2	Beatles	Bond	bud	cat		Doors	L3

그림 2. 디렉토리 노드의 배열. 저장되는 키는 여러 개의 노드에 키 값의 순서에 맞게 배열된다.

1 번 노드의 배열이 그림 2와 같다고 가정하면, 'Beach'는 배열의 네 번 째 방에 넣어진다. 마찬가지로 만일 'book'이라는 키는 2번 노드 배열의 2번 방에 들어가야 된다. 그런데 'bud'가 들어있으므로 'Bond' 우측의 key들을 한 칸씩 우측으로 밀고 저장한다. 그런데 2번 노드의 배열은 이미 다 차서 만원인 상태이므로 1023번 방의 데이터는 배열 밖으로 밀려나게 된다. 이렇게 밀려나는 데이터는 MI 상의 좌, 우 노드로 보내지게 된다.

이 때, 새로 삽입되는 키의 방 번호가 0~511의 범위이면 삽입점 이전의 방들을 좌측으로 이동한 후, 0번 방의 키 값을 좌측 링크가 가리키는 디렉토리 노드에 옮기고, 512~1023 범위이면 우측으로 이동하여 우측 끝 데이터를 우측 노드의 0번 방에 옮긴다. 이러한 배열의 이동 과정은 한 디렉토리 노드 내에서 최대 1024개의 방 데이터를 이동해야 하지만, 모든 연산이 메모리에서 수행되므로 순식간에 끝나게 된다.

만일 옮겨 갈 노드도 이미 만원인 상태라면 두 노드 사이에 새로운 노드를 삽입하고 이 노드에 이 데이터를 넣는다. 이렇게 새로 삽입된 노드의 좌 우의 노드들은 이미 배열이 꽉 찬 상태이므로 좌측 노드의 하위 1/3, 우측 노드의 상위 1/3의 배열 데이터를 옮겨 받는다. 이러한 분배는 만원으로 인한 데이터의 잦은 이동을 방지해 주고 디렉토리 노드들 간의 부하 균형을 제공한다.

새로 삽입된 노드는 좌, 우, 상, 하의 노드들과 주소 값을 주고 받아 링크를 갱신한다. 깊이가 같은 노드들의 개수가 증가하여 128을 초과하는 경우 깊이를 증가시킴으로써 한 노드가 관리하는 하위 노드의 수를 128개로 제한하였다.

디렉토리 노드는 단일 실패점이 되므로 이의 보완책으로 각 디렉토리 노드들은 좌우의 노드들과 자신들의 리스트 배열 내용을 주고 받아 복제한다. 또, 지역별로 디렉토리 그룹의 복제물을 둠으로써 확장성 및 안전성을 향상시킨다.

3.3 마스터 인덱스의 관리

앞 절에서 언급했듯이 MI는 트리에서 깊이가 1인 128개의 디렉토리 노드들의 인덱스 값과 IP 주소 값으로 구성된다. 이 시스템에서는 모든 노드가 MI를 갖고 있으므로, 변동이 발생하면 그 내역은 즉시 전 노드에 통지된다. 변하는 한 개의 인덱스와 IP 값의 크기를 합하면 64 바이트 밖에 되지 않으므로 브로드캐스트하는 데에는 무리가 없지만, 브로드캐스트 횟수를 줄이기 위하여 다음과 같이 개선하였다.

① 노드 8개를 한 개의 블럭으로 묶어 128개를 16개의 블럭으로 나눈다.
② 블록 내의 노드들은 가장 좌측의 리더 노드와 IP 및 인덱스 값을 주고 받는다.
③ MI를 16개의 리더 노드들 값만으로 구성한다.
④ 리더 노드들의 변경 시에만 그 내역을 전체 노드들에게 통보한다.

블로킹을 함으로써 MI 크기가 1/8로 줄어든다. 128개 대신 16개의 리더 노드 값만 통지되므로 통지 빈도도 1/8로 줄어든다. 검색 시에는 검색어들이 처음 16개의 리더 노드에 집중되지만 짧은 시간 내에 128개의 이상의 디렉토리 노드들로 분산된다.

처음에 이웃 노드로부터 MI를 얻는 과정은 불가피 하지만 Chord[1]와 같은 경우도 검색 시 해당 식별 영역에 등록된 노드를 찾기 위해서는 매번 이러한 과정을 거친다.

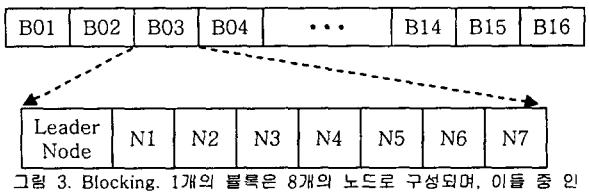

그림 3. Blocking. 1개의 블록은 8개의 노드로 구성되며, 이들 중 인덱스 값이 가장 낮은 노드가 리더 노드가 되어 다른 7개 노드의 IP와 인덱스를 관리한다.

4. 검색

검색을 위해서는 우선 해당 노드에 액세스해야 한다. 이를 위해

서는 MI를 이용하여 검색어가 어떤 블록에 존재하는지를 알아낸 후, 그 블록의 리더 노드와 통신함으로써 노드를 찾는다.
데이터의 검색 시는 저장할 때와 같은 방식으로 검색어를 배열 값과 비교한다. 비교할 문자열이 'a'나 'the'로 시작하는 경우 검색어에서는 잘라내지 않고 배열 값에서만 잘라낸 후 비교한다. 해당 노드를 찾아 1024개의 데이터 모두를 이진 탐색하는 경우 최대 10번 이내에 검색이 완료 된다.
이러한 메커니즘은 검색어를 만족하는 첫 번 째 키를 색인에 의해 직접 검색할 수 있게 해 주며 검색어를 만족하는 키가 하나 이상인 경우 그 키는 반드시 처음 검색된 키 바로 뒤에 순차적으로 존재하므로 다음 키들을 확인함으로써 만족하는 키의 집합을 모두 얻어낼 수 있게 해 준다. 그러므로 파일명 앞에 '[e-book]' 같은 키워드를 붙이는 경우, 같은 키워드를 갖는 데이터끼리 이웃하여 한두 개의 노드에 모여 있게 되므로 추가로 다른 노드와 통신을 하지 않고도 검색이 순식간에 행해진다. 검색 결과는 MS/DOS의 dir 명령의 결과와 유사하다. 예를 들어, 'be'라는 검색어를 보내면, 'Beach'와 'Beatles' 등 'be'로 시작하는 모든 키 값을 검색한다. 그림 1, 2와 같은 시스템에 'a'라는 검색어를 보낸다면 이 프로토콜은 0번 노드의 모든 키 값과 1번 노드에서 'a'로 시작하는 'Andy'와 'Art'를 알파벳 순으로 보여 준다. 이 때 0번 노드의 링크는 1번 노드의 주소를 가리키고 있으므로, 0번 노드의 배열 리스트의 가장 높은 값이 검색어에 대해 'true' 반응을 보일 경우, 링크된 1번 노드가 검색어를 넘겨받아 검색을 계속한다.
일반적으로 키워드 검색은 여러 개의 노드들로부터 응답을 받아야 하므로 시간이 걸리지만 이 시스템에서는 처음 찾은 한 개의 노드로부터 검색어를 만족하는 모든 결과를 얻을 수 있고 경우에 따라 한 개의 링크 노드에 추가로 접근한다.
만일 저장된 키의 개수가 K개라면 액세스에 필요한 스텝 수는 항상 $\log_{128}(K/1024)+1$ 스텝을 넘지 않으며, 1000만 개의 키 데이터인 경우 세 단계 이내에 검색이 완료된다. 구조적 P2P 시스템인 Chord[1]의 경우 전체 검색 스텝 수가 O(log N)으로 노드의 개수 N과 관계가 있으나 여기서는 데이터의 개수와 관계가 있다.

5. 토론

A. 노드의 참여
새로 참여하는 모든 노드는 디렉토리 노드가 아니므로 MI를 관리하는 루트 노드가 된다. 그러므로 이웃 노드로부터 MI를 구하여 루트노드가 된다.

B. 노드의 이탈
디렉토리 노드가 아닌 노드의 이탈은 문제가 되지 않는다. 이 노드가 이미 등록한 키 값은 삭제를 하지 않는다. 이런 경우 등록된 키의 실제 노드가 존재하지 않는 불일치가 발생하지만 삭제에는 시간이 걸리고, 삭제하면 나중에 다시 참여할 때 재 등록을 해야 하므로 바로 이탈한다. 불일치 상황이 길어지면 별도의 쓰레기 수집 과정을 통하여 삭제한다.

C. 디렉토리 노드의 이탈
디렉토리 노드가 이탈할 경우에는 좌우의 링크 노드들에게 통보하고 탈퇴한다. 디렉토리 노드의 배열 리스트는 좌, 우의 노드에서 복제하여 관리하기 때문에 이 노드들은 공백이 된 곳에 새로운 노드를 삽입하고 필요한 자료를 넘겨준다. 새로운 노드는 자기의 상하, 좌우 노드들과 링크의 갱신 등 필요한 절차를 거친다. 익명성 있는 일반노드가 디렉토리 노드로 선정되므로 이탈은 일어 날 수 있다. 그러나 디렉토리 노드들의 잦은 이탈은 바람직하지 않으므로 접속 상황에 대한 통계치를 분석하여 상황이 더 나은 노드가 발견되면 이로 교체하는 것이 바람직하다. 특히 다이얼 업(Dial-up) 방식으로 연결된 노드는 피해야 한다.

D. 노드의 실패
디렉토리 노드의 실패는 이탈을 할 때와는 다르다. 왜냐하면 이웃 링크 노드에 통지를 하지 않기 때문이다. 이에 대비하여 디렉토리 노드들은 이웃 링크 노드들과 주기적으로 얼라이브 메시지를 주고 받는다. 얼라이브 메시지 수신이 타임아웃 한계를 넘기면 확인 절차를 거치고 실패로 인식되는 경우 이탈의 경우와 같이 처리한다.

D. 동기화
다운로드 된 파일은 공유 폴더에 저장되면서 자동적으로 등록된다. 그러나 사용자가 파일을 추가하거나 삭제하는 경우 이러한 변동 내용이 디렉토리 노드에 반영되어야 한다. 이를 위하여 실제 파일 내역과 저장된 파일 리스트의 스냅샷을 주기적으로 비교 한다.
파일을 추가한 경우는 통상의 등록 절차를 거치면 된다. 그러나 삭제된 리스트는 등록과 같은 절차를 거치되, 검색어의 조작 식별자를 'd'로 하여 노드 배열에서 찾은 다음 건수가 복수인 경우 자신의 주소를 삭제하고 자신 밖에 없으면 방의 내용을 삭제한다.
변동내역이 모두 반영되면 새로운 스냅샷을 취한다.

6. 결론 및 향후 연구과제

본 논문에서는 검색 키워드를 만족하는 데이터의 집합을 단 한 번의 액세스로 얻을 수 있게 해 주는 새로운 혼합 방식의 P2P 검색 기법을 제안한다. 제안된 기법은 하나 이상의 검색결과를 얻는 키워드 검색을 구조적인 P2P 시스템과 동등한 속도로 가능하게 한다.
제안된 기법은 다른 여러 분야에서도 응용될 수 있는데, 예를 들면 전문 검색 사이트나 신문사의 편집국처럼 키워드 검색을 많이 필요로 하는 회사들, 그리고 협업을 하는 연구기관들이나 파일 공유를 필요로 하는 일반회사들에서 서로 공유할 파일들을 가상적으로 한 곳에 모음으로써 파일 서버를 운영하는 효과를 볼 수 있는 등, 많은 활용분야를 기대할 수 있다.
향후 연구과제는 제안된 기법에서 다양한 환경 변수를 고려하여 기존의 검색 기법과 비교하여 종합적인 성능 분석을 수행할 것이다.

참고 문헌

[1] Stoica, I., Morris, R., Karger, D., Kaashoek, F., and Balakrishnan, H. Chord: A scalable Peer-to-Peer Lookup Service for Internet Applications. ACM SIGGCOM 2001(San Diego, CA, Aug27-31, 2001)

[2] Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S. A Scalable Content-Addressable Network SIGCOMM'01 (San Diego, CA, Aug27-31, 2001)

[3] Rowstron, A., Druschel, P. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer Systems. Proceeding of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Heidelberg, Germany, 2001)

[4] Chawathe, Y., Ratnasmy, S., Breslau, L., Lanham, N., Shenker, S. Making Gnutella-like P2P Systems Scalable. ACM SIGCOMM 2003(Aug 25-29, 2003)

[5] Tsoumakos, D., Rossopoulos, N. Analysis and Comparison of P2P Search Methods. CS-TR-4539, UMIACS-TR-2003-107

[6] http://www.napster.com, Napster website

[7] http://www.gnutella.com, Gnutella website

[8] http://www.webopedia.com/TERM/S/server_farm.html