효율적인 키워드 검색을 지원하기 위한 계층적 확장 Chord

이승은°, 진명희, 김경석†
부산대학교 컴퓨터공학과°, 부산대학교 정보컴퓨터공학부†
{selee°, mhjin, gimgs}@asadal.pusan.ac.kr

Hierarchical extended Chord to Support Efficient Keyword Search

Seung-eun Lee°, Myoung-hee Jin, Kyong-sok Kim†
Dept. of Computer Engineering. Pusan National University°
Division of Computer Science and Engineering. Pusan National† University†

요 약

현재 Peer-to-Peer (P2P) 파일 공유 시스템은 특정 데이터 항목을 저장한 노드의 위치를 어떻게 효율적으로 찾을 것인가에 대해 연구되고 있다. DHT 기반 구조는 확장 가능하며 정확한 매치(exact-match) 질의가 가능하지만, 정확한 매치가 아닌 질의에 대해서는 효율적이지 못하다. 본 논문은 DHT 기반의 P2P 파일 공유 시스템에서 확장 가능한 키워드 기반 파일 검색을 제공하기 위한 메커니즘을 제안한다. 우리의 제안된 구조는 높은 능력을 가진 "슈퍼피어"를 둠으로써 많은 파일을 가진 공통 키워드로 인해 발생하는 과도한 저장 공간 소비와 네트워크 트래픽을 감소시킨다.

1. 서 론

P2P 시스템의 근간에는 분산된 노드들로 구성된 P2P 네트워크가 일반적으로 응용 프로그램 계층에서 생성되는 오버레이(overlay) 네트워크가 있으며, 이에 관한 연구들은 초기의 Napster[1]와 같은 중앙집중식 연결을 지나서 완전하게 분산된 네트워크 연결에 집중되고 있다. 이런 오버레이 네트워크는 Gnutella[2] 등과 같은 비구조적 P2P 네트워크와 분산해쉬테이블(distributed hash table : DHT) 기반으로 구현되는 구조적 P2P 네트워크로 분류된다. 전자의 경우 확장 가능성(scalability)에 제한이 있는 반면, 구조적 P2P 네트워크는 높은 분산으로 인한 확장 가능성(scalability)과 네트워크 고장에 대한 신뢰성 (reliability)을 향상시킬 수 있으며 N 노드가 존재하는 시스템에서 자원 탐색을 위해 O(logN) 홉(hop)의 효율적인 탐색 또한 가능하다. 구조적 P2P 네트워크의 Chord[3], CAN[4], Pastry[5]와 Tapestry[6]와 같은 프로토콜은 DHT 시스템으로 인해 단지 정확한 매치(exact-match) 질의만 가능하며, 노드의 컴퓨팅 능력이 같다고 가정하고 있다.

하지만, 실제 현실에서는 노드들의 능력이 서로 다르므로 P2P 네트워크를 고정 네트워크로 가정하여 네트워크 대역폭과 참여 시간으로 노드의 컴퓨팅 능력을 결정하며, 컴퓨팅 능력에 따라 관리하는 인덱스 정보의 크기를 불균등하게 할당하여 성능이 증가하는 방안을 제안한다.

본 논문은 노드의 능력을 고려한 DHT 기반의 P2P 파일 공유 시스템에서 확장 가능한 키워드 기반 파일 검색을 제안한다. 성능이 높은 피어(슈퍼피어)에게 매우 많은 파일과 관련되어 있는 공통 키워드에 대한 인덱스를 관리하게 함으로써 과도한 저장 공간 소비와 네트워크 트래픽을 줄인다.

2. 관련 연구

이번 장에서는 DHT를 사용하는 Chord를 사용하여 키워드 검색을 지원하기 위해 제안된 역 인덱스 분산해쉬 테이블(inverted distributed hash table)을 사용한 연구에 대해 살펴본다.

2.1 DHT 기반 P2P 시스템에서의 키워드 검색

DHT 기반 P2P 시스템에서 특정 파일을 포함하는 노드를 찾는 것은 DHT에 저장된 <file ID, value>로 대응시킴으로써 간단히 찾을 수 있다. Chord[3] 같은 논리적 오버레이 구조 안으로 노드를 참여시키는 DHT 기반 P2P 시스템은 자원 탐색을 위해 O(logN) 홉(hop)의 효율적인 탐색이 가능한 반면, 파일은 유일한 ID를 통해 위치가 정해지므로 단순히 정확한 매치(exact-match) 질의만 가능하다.

이러한 DHT 기반 P2P 시스템에서 키워드 기반 질의를 지원하기 위해 기존의 Chord[3]를 확장한다. 확장 Chord는 <file ID, value> 대신에 각 DHT 노드에서 <keyword, list of values> 정보를 유지하여 검색을 가능하게 한다. 'list of values'는 키워드와 관련된 모든 파일을 실제로 저장하고 있는 노드의 IP 주소를 포함한다. 사용자가 키워드와 관련된 파일들에 대해 검색 할 때, 키워드를 해쉬한 값과 대응되는 노드로 질의를 전달하여 파일들의 위치를 얻는다. 만약 사용자가 다중 키워드로 파일을 찾는다면, 각 키워드에 대한 결과를 교차(intersection) 함으로써 최종 결과를 얻는다.[7]

확장 Chord의 질의 처리는 단순하게 구성되는 반면, 몇 가지 단점이 있다. 첫째, <keyword, list of values> 대응으로 DHT 노드들의 저장 공간이 불균등하게 된다. 이런 불균등 현상은 일부 키워드가 많은 수의 파일들과 공통으로 관련되기 때문에 일부 노드가 과도한 저장 공간을 요구하게 된다. 둘째, 매우 긴 리스트의 파일정보와 관련된 공통 키워드들을 포함하는 검색 질의는 네트워크 트래픽의 양을 대량으로 증가시킨다.

이런 확장 Chord에서의 두 가지 단점을 해결하기 위해 Keyword Fusion[8] 메커니즘이 제안되었다.

2.2 Keyword Fusion 메커니즘

P2P 파일 공유에서 확장 가능한 키워드 기반 파일 검색을 효율적으로 지원하기 위해 키워드 통합 메커니즘[8]인 Keyword Fusion 이라 불리는 제안된 구조는 FD(Fusion Dictionary)라는 공통 키워드를 관리하는 분산 자료 구조를 두어 공통 키워드로 인해 과도한 저장 공간이 소비되는 피어들에 대한 부담을 없애며, 검색 조건에 맞추어 사용자의 질의를 전환함으로써 네트워크 대역폭 소모를 줄인다. 그러나 이 메커니즘은 시스템에 존재하는 모든 노드가 공통 키워드에 대한 인덱스 정보를 일관되게 유지해야 하므로 유지가 어렵고, 파일 인기와 저장 공간에 따라 공통 키워드를 판단하는 임계값(threshold value)이 빈번하게 변하므로 시스템 전체 노드의 인덱스 유지비용이 큰 단점이 있다.

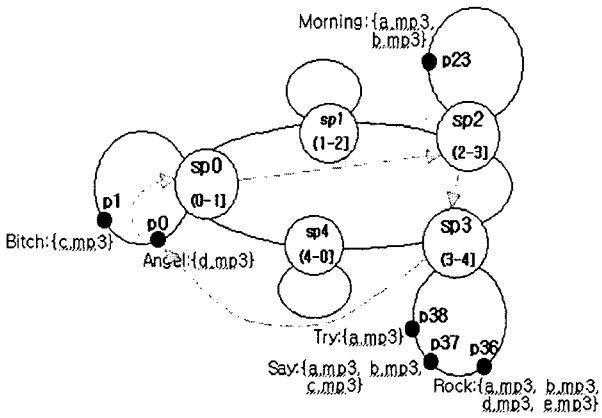
다음 장에서 Keyword Fusion[8]의 확장 가능한 키워드 검색에 "슈퍼피어" 사용으로 인한 하이브리드(hybrid) 접근 방법을 접목시킴으로써, Keyword Fusion[8] 구조의 단점을 보완하는 계층적 확장 Chord 시스템 구조를 제안한다.

3. 시스템 구조

우리는 키워드 검색 능력을 지원하기 위하여 계층적 확장 Chord 시스템 구조를 제안한다. N개의 피어를 가진 주어진 P2P 시스템에서 제안된 확장 DHT 메커니즘을 사용하여 식별자 공간(identifier space)인 하위 레벨 오버레이 네트워크를 구성하고, 이 N개 피어들 중의 K개만큼 슈퍼피어를 선택하여 상위 레벨 오버레이 네트워크를 구성한다. 하위 레벨 오버레이 네트워크는 일정한 범위로 구분되어 각 슈퍼피어가 관리하게 된다.

3.1 계층적 확장 Chord 시스템 구조

[그림 1]은 우리 시스템의 기초적인 구조를 나타낸다. 우리는 확장 Chord 시스템에서 저장 공간 소비와 네트워크 트래픽에 대한 결정을 발생시키는 공통 키워드 - 많은 파일에 공통으로 관련되어 저장 공간이 크며 인기가 있는 키워드 - 를 효율적으로 관리하기 위해 슈퍼피어를 가지는 계층적인 확장 Chord 시스템을 제안한다.



[그림 1] 키워드 기반 검색에 대한 확장 Chord

File ID	Keywords
a.mp3	Morning, Rock, Say, Try
b.mp3	Morning, Music, Rock, Say
c.mp3	Bitch, Say, Music
d.mp3	Angel, Rock, Music
e.mp3	Rock, Music

[표 1] 키워드 기반 검색에 대한 확장 Chord

superpeer	range
sp0	(0 - 1]
sp1	(1 - 2]
sp2	(2 - 3]
sp3	(3 - 4]
sp4	(4 - 0]

[표 2] 슈퍼피어 테이블

peer	identity
p0	p0's IP
p1	p1's IP

[표 3] sp0의 피어 테이블

제안된 구조에서 각각의 슈퍼피어는 다음의 세 가지 정보를 유지한다.

1) 슈퍼피어가 관리하는 범위에 포함된 피어들의 주소 (표 3),
2) 슈퍼피어와 관리하는 범위 사이의 대응(표 2)
3) CKI(Common Keyword Index)

피어가 질의에 포함된 키워드와 관련된 파일을 요청할 때, 피어는 자신의 슈퍼피어에게 직접 이 요청을 보낸다. 그러면 슈퍼피어는 자신의 CKI를 참조하여 질의에 포함된 키워드가 공통 키워드인지 확인한다. CKI에 속한 키워드가 있다면 공통 키워드를 제외한 새로운 질의를 생성하며 처음 질의는 질의 바디에 포함된다. 질의요청 피어의 슈퍼피어가 수정된 질의의 키워드(일반 키워드)를 포함하고 있는 범위를 관리한다면, 그 슈퍼피어는 키워드를 관리하는 피어 테이블의 ID의 successor를 탐색하고 그 결과를 반환한다. 그렇지 않으면, 그 슈퍼피어는 키워드가 포함된 범위를 관리하는 슈퍼피어에게 질의를 전달한다. 응답할 책임이 있는 슈퍼피어는 자신의 피어 테이블에 있는 successor 피어를 탐색하고 그 결과를 처음 요청을 한 피어에게 반환한다.

3.2 공통 키워드 인덱스와 부분 키워드 리스트

슈퍼피어가 관리하는 CKI(Common Keyword Index)는 공통 키워드를 유지하는 분산 자료 구조로서, 과도하게 저장 공간을 쓰는 피어들의 로드(load)를 줄이며, 공통 키워드를 피함으로써 사용자의 질의를 효율적으로 처리한다. CKI는 DHT 노드의 저장 공간 소비가 과도하다고 생각될 때 사용하는 것으로, CKI에 포함된 공통 키워드인지 확인하고 포함되지 않은 키워드일 경우 CKI에 키워드를 등록한 후 키워드와 관련된 파일의 IP 주소를 제거한다. 즉, CKI는 'list of values'에 저장하는 키워드와 관련된 파일의 IP 주소 없이 <keyword>로 저장된다.

이처럼 키워드와 관련된 파일의 IP 주소가 삭제된 공통 키워드를 포함하는 질의를 다루기 위한 메커니즘으로 PKL(Partial Keyword List)라는 데이터 구조를 제안한다. PKL은 파일의 키워드들과 CKI를 비교하여 교집합한 결과로써 각 파일마다 교집합된 공통 키워드 PKL이 저장된다.

PKL을 사용한 키워드 검색을 [그림 1]로써 살펴보자. [그림 1]에서 우리는 Music이 CKI에 포함된 공통 키워드이고 Morning과 Say가 일반 키워드라고 가정한다. 만약 p0에서 사용자가 "Music AND Morning AND Say"라는 질의를 발행할 경우, p0는 자신을 관리하는 슈퍼피어 sp0에게 질의를 보낸다. 질의를 받은 sp0는 자신의 CKI를 검색하여 Music이 공통 키워드임을 발견한 후, 일반 키워드로 구성된 "Morning AND Say"로 질의를 수정하고 처음 질의는 질의 바디에 포함한다. 그 후, sp0는 슈퍼피어 테이블(표 2)을 확인하여 Morning 키워드를 해쉬한 값(ID)이 포함된 범위를 관리하는 슈퍼피어 sp2를 발견하고 질의를 보낸다. sp2는 자신의 피어테이블에서 Morning 키워드 ID의 successor를 탐색하고 successor가 관리하는 Morning 키워드와 관련된 파일인 {a.mp3, b.mp3}를 Say 키워드 ID를 포함하는 범위를 관리하는 슈퍼피어 sp3에게 질의와 함께 보낸다. Say 키워드의 {a.mp3, b.mp3, c.mp3}와 Morning 키워드의 {a.mp3, b.mp3}를 교차하여 일반 키워드가 모두 포함된 파일 {a.mp3, b.mp3}를 처음 질의를 요청한 p0에게 보낸다. p0는 받은 파일들("Morning&Say" 질의 결과 파일들) 중에서 질의 수정과정에서 제외되었던 공통 키워드 Music을 포함하는 파일("Music AND Morning AND Say" 질의 결과 파일들)을 최종 결과로 얻기 위해, 각 파일의 PKL을 참조하게 된다. a.mp3 파일의 PKL은 a,mp3의 키워드들과 CKI를 교집합하여 나온 결과가 없으므로 비어있으며, b.mp3 파일의 PKL은 Music이 포함되어 있다. 그러므로 Music이 포함된 처음 질의의 결과로 {b.mp3}를 얻는다.

3.3 시스템 관리

1) 부트스트래핑(Bootstrapping) : 처음 K개 피어들이 시스템에 참여할 때 이 피어들은 슈퍼피어가 되어 상위 오버레이 네트워크를 구성하며, 시스템에 추가적으로 피어들이 참여할 때는 Chord 방식으로 하위 네트워크를 구성하고 직접적으로 인접한 이웃피어로부터 자신의 슈퍼피어를 발견하게 된다. 만약 슈퍼피어의 컴퓨팅 능력이 참여하는 피어의 컴퓨팅 능력보다 현저히 낮다면 슈퍼피어는 참여하는 피어로 대체되며, 기존 슈퍼피어는 일반노드가 되어 확장 Chord를 구성하게 된다.

2) 부하 균형 (Load Balancing) : 파일의 인기는 불균등하기 때문에 특정 범위가 다른 것 보다 더 많은 요청을 경험할 수 있다. 이러한 경우, 슈퍼피어 사이의 부하(load)를 조절하거나 슈퍼피어의 수를 변경함으로써 부하 균형을 유지한다. P2P에서는 노드의 삽입과 삭제가 빈번히 일어나므로 슈퍼피어가 관리하는 범위를 동적으로 결정해야 한다. 우리는 각각의 슈퍼피어가 자신의 최대 수용능력 M을 안다고 가정할 경우, 노드 삽입이 자주 발생해서 슈퍼피어가 관리하는 범위의 노드 수가 M 이상이 된다면 범위를 분할하여 충분한 여분의 수용능력을 가지고 있는 이웃 슈퍼피어 또는 참여하는 피어 중에서 선정된 새로운 슈퍼피어에게 분할된 범위를 할당한다. 노드의 삭제가 많아서 이웃 슈퍼피어 범위에 있는 노드 수와 자기 범위의 노드 수를 합쳐 M 이하가 된다면 노드의 합병이 발생하여 슈퍼피어 1개를 제거한다.

3) 장애(failure) 발견 및 관리 : 일반 피어 장애일 경우는 Chord[3] 시스템과 동일하며, 슈퍼피어 장애일 경우에는 주기적인 keep-alive 메시지를 통하여 그것의 이웃에 의해 발견된다. 슈퍼피어 장애가 발생할 경우, 장애가 발생한 슈퍼피어에 의해 관리되던 범위 내의 모든 피어들은 하나 또는 그 이상의 새로운 슈퍼피어에게 할당 되고, 새로운 슈퍼피어는 갱신된 슈퍼피어 정보를 모든 슈퍼피어에게 알리며 각각의 슈퍼피어들은 자신의 정보를 갱신한다. 예기치 못한 슈퍼피어 장애 발생일 경우, 관리 범위의 피어 정보를 잃어버리는 것을 대비하여 슈퍼피어는 자신의 이웃 k개 피어에게 피어 정보를 복사한다.[1] 정보의 일관성을 유지하기 위하여 관리하는 피어 정보가 갱신될 때마다 이웃 피어에게 알린다.

4. 결론 및 향후 연구과제

우리는 DHT 기반 P2P 시스템에서 확장 가능한 키워드 기반 파일 검색을 제공하는 메커니즘을 제안하였다. 확장 Chord 시스템에 컴퓨팅 능력이 뛰어난 노드를 슈퍼피어로 두어 검색 비용을 감소시키며, 많은 수의 파일을 가지는 공통 키워드 인덱스를 효율적으로 관리하게 함으로써 네트워크 트래픽과 저장 공간 소비를 균등하게 분산시킨다.

현재, 슈퍼피어가 관리하는 범위를 관심영역으로 분류하는 방법과 노드의 능력을 고려한 슈퍼피어 변동에 있어 더욱 안정적인 시스템을 지원할 방안을 고안 중이며, 본 논문에서 제안한 초기 메커니즘의 성능을 평가하기 위하여 실험단계에 있다.

참조문헌

[1] Napster, http://www.napster.com/
[2] Gnutella, http://www.gnutella.com/
[3] I. Stoica, R. Morris, D. Karger, M.F Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications", ACM SIGCOMM'01, Aug. 2001
[4] S. Ratnasamy, P. Francis, M. Handley, R. Karp, "A Scalable Content-Addressable Network", ACM SIGCOMM'01, Aug. 2001
[5] A. Rowstron, P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale Peer-to-Peer systems". IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), pp. 329-350, November 2001
[6] B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for faulttolerant wide-area location and routing," Tech. Rep. UCB/CSD-01-1141, Computer Science Division, University of California, Berkeley, Apr 2001
[7] P. Reynolds and A. Vahdat, "Efficient Peer-to-Peer Keyword Searching," In Proc. of Middleware. 2003, Rio de Janeiro, Brazil, 2003.
[8] L. Liu, K. D. Ryu, and K. W. Lee, "Keyword Fusion to Support Efficient Keyword-based Search in Peer-to-Peer File Sharing", In 4th Int'l Workshop on Global and P2P Computing, Chicago IL, April 2004.

1) k는 2 또는 3의 작은 값으로도 충분하다.