Lookup 효율성을 위해 Topology를 고려한 Chord 시스템

차봉관[O] 한동윤 손영성 김경석
부산대학교 컴퓨터공학과
{bgcha[O], dyhan, ysson, gim2005}@asadal.cs.pusan.ac.kr

Topology-based Chord system for lookup services

Bonggwan Cha[O], Dongyun Han, Youngsung Son, Kyongsok Kim
Dept. of Computer Science and Engineering, Electronics and Telecommunications Research institute, Division of Computer Science and Engineering, Pusan National University

요 약

최근에 P2P(Peer-to-Peer) 시스템에서 효율적인 자원 탐색 방법에 대해 많이 연구되고 있다. P2P overlay network 중 하나인 Chord는 자원을 효율적으로 탐색할 수 있는 간단한 P2P 프로토콜이다. 대부분의 P2P 시스템은 overlay network를 형성하므로 노드와 노드 사이의 물리적인 거리를 고려하지 않으므로 서로 이웃한 노드라도 실제 물리적인 latency가 클 수 있다는 문제점을 가지고 있다. 이 논문은 물리적으로 가까운 노드들을 하나의 subnet으로 만들어서 물리적인 latency를 감소시키고 각 subnet안에 복사본(replica)을 둠으로써 탐색(lookup)의 효율성을 향상시킨다.

1. 서 론

P2P 시스템은 P2P 시스템에 참여하는 노드들의 자원을 공유하는 분산 시스템으로 여기에 참여하는 노드들은 서버와 클라이언트의 역할을 모두 수행한다. 대부분의 P2P 시스템의 가장 핵심이 되는 기능은 데이터를 효율적으로 저장하는 일이다. 현재 가장 잘 알려진 P2P 시스템으로는 파일 공유시스템인 Napster[1]와 Gnutella[2]가 있으며, 이 시스템들은 각 노드에 파일을 저장하고 파일을 원하는 노드는 시스템에 속한 그 파일을 저장하는 노드로부터 직접 파일을 전송 받는다. 대부분의 P2P 시스템의 가장 중요한 기능은 데이터를 효율적으로 배치하고 탐색(lookup)하는 것이다. 그러나 Napster나 Gnutella의 경우 확장성(scalability) 문제를 가지고 있다.

확장성 문제를 해결하기 위해 확장 가능한 P2P overlay network(CAN[3], Chord[4], Pastry[5], Tapestry[6]) 들이 제안되고 있다. P2P overlay network은 데이터의 배치와 탐색 알고리즘을 제공한다. 즉, 데이터들이 저장되어 있는 위치 정보들이 overlay network상의 노드들에 분산 배치되기 때문에 새로운 데이터를 저장하기 위해서는 어떤 노드에 저장할 것인지를 결정하는 방법과 어떤 데이터가 저장된 위치를 알기 위해 데이터의 위치 정보를 가지고 있는 노드를 찾기 위한 탐색 알고리즘을 제안하고 있다.

P2P overlay network 에서는 노드의 IP주소를 해쉬하여 나온 결과 값에 따라 시스템에 조인되므로 노드를 사이의 물리적인 거리를 고려하지 않는다. 즉 가상의 network에서 한 홉(hop)이라도 실제 여러 topology를 경유할 수 있기 때문에 시스템의 성능을 감소시키는 요인이 된다. 이 논문은 물리적인 거리를 측정하여 서로 인접한 노드들을 모아 하나의 subnet으로 구성해 물리적인 latency를 줄일 수가 있다. 또한 Replica를 이용하여 자신의 속해있는 subnet에서 원하는 데이터를 찾는 확률을 높여서 탐색의 효율성을 향상 시킨다.

이 논문의 구조는 다음과 같다. 2절에서 Chord와 Grapes에 대한 간략한 소개를 하고, 3절에서는 우리가 제안하는 시스템의 구조와 기능에 대하여 설명한다. 마지막으로 4절에서는 결론과 향후 연구 과제를 제시하고 있다.

2. 관련 연구

2.1 Chord

Chord 시스템은 P2P 응용을 위한 분산처리 자원탐색 프로토콜로, 분산 탐색을 지원하며 해쉬 함수를 이용하여 데이터의 삽입과 탐색을 수행한다. Chord는 2^k 크기의 원형 식별자 공간을 사용하며 각 노드는 IP주소를 SHA-1과 같은 해쉬 함수로 해쉬하여 nodeID를 구한 다음 원형 식별자 공간의 해당 nodeID에 위치한다. 데이터의 위치 정보는 (key, value) 쌍으로 표현되며 데이터가 저장될 노드의 위치는 key를 해쉬한 값에 의해 정해진다. 원형 식별자 공간에서 각 노드는 successor, predecessor의 정보를 유지하여 링을 형성하고 노드가 fail되었을 때 시스템을 복원하기 위해 successor list의 정보를 유지한다.

Chord에서 key를 해쉬한 값을 식별자 공간에 대응시킬 때 원형 식별자 공간에서 해쉬된 키 값과 같은 nodeID를 가지는 노드에 저장하거나 같은 nodeID를 가진 노드에 저장하거나, 같은 nodeID를 가진 노드가 없을 때는 바로 뒤의 노드에 저장한다. 이 노드를 successor 노드라 부른다. 각 노드는 전체 네트워크 상의 노드에 대한 정보를 분

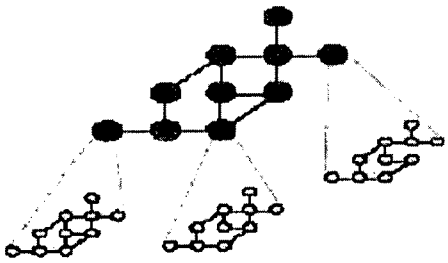

그림 1. Grapes의 구조

산된 동적 환경에서 효과적으로 만족시키기 위해 finger table을 유지한다. Finger table은 데이터를 삽입하거나 데이터를 관리하는 노드를 찾기 위해 lookup 메시지를 해당 노드에게 전달하기 위해 이용된다.

2.2 Grapes

Grapes[7]는 Topology를 고려한 P2P시스템이다. Grapes는 leader들로 구성되는 super-network와 물리적으로 가까운 노드들로 구성된 sub-network으로 이루진 시스템이다. 조인을 원하는 노드는 bootstrap node와 물리적 거리를 측정하여 그 값이 threshold 값보다 작으면 bootstrap node에 속하는 sub-network에 조인하고 threshold 값보다 크면 super-network의 leader와 물리적인 거리를 측정하여 threshold 값보다 작은 leader의 sub-network에 조인된다. 만약 모든 leader와 물리적 거리를 측정한 값이 Threshold 값보다 크면 새로운 노드는 super-network에 조인되고 sub-network의 leader가 된다. Sub-network의 모든 노드는 다른 sub-network의 데이터를 탐색하기를 원할 때 자신의 sub-network에 속해 있는 leader를 통해서만 가능하기 때문에 leader에 대한 부담 크면 leader가 fail될 때 leader의 교체하는 비용 크다. 또한 각 sub-network의 크기가 동일하지 않으므로 node의 수가 적은 sub-network의 노드는 데이터를 탐색하기 위해 다른 sub-network에서 데이터를 찾는 확률이 높아 탐색의 효율성이 떨어진다. 우리가 제안하는 논문은 subnet의 사이즈를 고정하여 subnet에 노드의 수를 균일하게 만들기 때문에 탐색의 효율성을 향상시킨다.

3. 시스템 구조

우리가 제안하는 P2P 시스템은 모든 노드로 이루어진 Global network와 물리적으로 가까운 노드들로 이루어진 여러 개의 subnet으로 이루어진 시스템이다. 각 노드는 Global network에 존재하고 그와 동시에 하나의 subnet에 속해 있다. 각 노드는 자신의 식별자를 가지기 위해 자신의 IP주소를 2개의 해쉬 함수(h1, h2)를 사용하여 global-nodeID와 local-nodeID를 구한다. 해쉬 함수는 SHA-1과 같은 해쉬 함수를 사용하며 Global-nodeID는 Global-network에서 데이터를 찾거나 노드를 조인할 때 사용되는 식별자이고, local-nodeID는 각 노드가 속해 있는 subnet에서 데이터를 찾고 노드를 조인할 때 사용되는 식별자이다

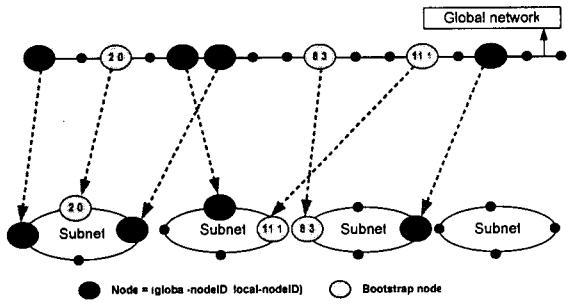

그림 2. 물리적 거리를 고려한 시스템 구조

물리적 거리의 측정 방법은 여러 가지가 있으나 우리는 ping를 통해서 값을 구하고 Global network과 subnet은 Chord lookup 알고리즘을 사용한다. 그림 1은 크기가 16인 원형 식별자 공간을 가지는 Global network에 크기가 4인 원형 식별자 공간을 가지는 4개의 subnet을 가지는 시스템의 구조이다. 각 노드는 global-network에 속해 있으며 또한 4개의 subnet 중 한 개의 subnet에 속해 있다. 각 subnet은 반드시 한 개의 bootstrap node를 가지며 각 노드는 Global network를 위한 finger tabal과 노드 fail되었을 때 시스템을 복원하기 위해서 successor list의 정보를 유지한다. 또한 노드는 자신이 속해 있는 subnet에 데이터 탐색을 위해서 필요한 finger table과 successor list의 정보를 유지한다. 2개의 finger table은 데이터 탐색과 삽입을 할 때 독립적으로 사용한다. Global-network에서는 global-nodeID만을 사용하고 subnet에서는 local-nodeID만을 사용한다. 각 subnet의 모든 노드는 자신의 속해 있는 subnet의 bootstrap node의 정보를 유지하고 Bootstrap node는 자신과 이웃하는 다른 subnet의 bootstrap node의 정보를 유지한다.

3.1 노드 조인

시스템에 조인을 원하는 새로운 노드는 시스템에 존재하는 bootstrap node의 정보를 반드시 알고 있다. 그림 3은 새로운 노드가 시스템에 조인(join)하는 과정을 보여주는 그림이다. 시스템에 조인을 원하는 새로운 노드는 자신의 IP주소를 2개의 해쉬 함수(h1, h2)를 이용하여 global-nodeID와 local-nodeID를 구한다. 먼저 global-nodeID를 사용하여 bootstrap node를 통해 Global network에 global-nodeID와 일치하는 위치에 조인하다. 그림 3에서 N1의 상태이다. 다음으로 새로운 노드는 ping 이용하여 bootstrap node와의 물리적 거리를 측정한다. 물리적 거리를 측정한 값이 threshold 값보다 작으면 그 bootstrap node가 속해 있는 subnet에 local-nodeID와 일치하는 위치에 조인한다. 만약 물리적 거리를 측정한 값이 threshold 값보다 크면 bootstrap node가 유지하는 다른 bootstrap node의 정보를 이용하여 그 bootstrap node와의 물리적 거리를 측정하여 threshold 값과 비교한다. Threshold 값보다 작으면 그 bootstrap node가 속해 있는 subnet에 조인한다. 그림 3에서 N2의 경우이다.

시스템에 있는 모든 bootstrap node와 물리적 거리를 측정한 값이 threshold 값보다 클 때는 2가지의 상황에

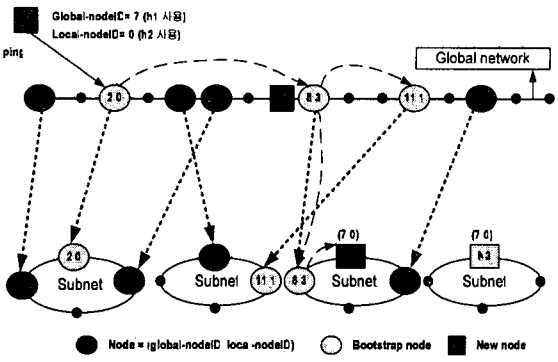

그림 3 노드의 조인

따라 다르게 조인된다. 첫 번째로 시스템이 가질 수 있는 subnet 수가 최대가 아닐 때는 시스템에 조인을 원하는 새로운 노드는 새로운 subnet을 가지고 그 subnet의 bootstrap node가 된다. 그림3에서 N3의 경우이다. 두 번째로 subnet이 시스템 가질 수 있는 최대의 subnet이 존재하면 조인을 원하는 새로운 노드는 threshold 값을 올려서 다시 조인을 시도한다.

마지막으로 Bootstrap node와 물리적 거리를 측정하여 threshold 값 보다 작아서 그 subnet에 조인할 때 만약 그 subnet에서 같은 local-nodeID를 가지는 노드가 존재할 경우에는 새로운 노드는 threshold 값을 올려서 다시 조인을 한다.

3.2 데이터 삽입

데이터의 삽입을 원하는 노드는 데이터의 key를 2개의 해쉬 함수(h1, h2)를 이용하여 global-nodeID와 local-nodeID를 구한다. 노드는 먼저 local-nodeID를 이용하여 데이터의 삽입을 원하는 노드의 subnet에서 local-nodeID를 관리하는 노드에 데이터를 삽입한다. 그리고 Global-nodeID를 이용하여 Global network에 global-nodeID를 관리하는 노드에게 데이터를 삽입한다.

데이터를 subnet과 Global network에 삽입하는 이유는 원하는 데이터를 탐색할 때 자신의 subnet에서 찾을 확률을 높여서 lookup의 효율성을 향상 시키기 위한 것이다.

3.3 데이터 삽입

데이터의 탐색을 원하는 노드는 데이터의 key를 2개의 해쉬 함수(h1, h2)를 사용하여 global-nodeID와 local-nodeID를 구한다. 데이터 탐색을 원하는 노드는 local-nodeID를 이용하여 자신이 속해 있는 subnet에서 local-nodeID를 관리하는 노드를 찾는다. 만약 원하는 데이터가 있으면 local-nodeID를 관리하는 노드에서 데이터를 받는다. 자신의 subnet에서 원하는 데이터가 없을 때는 global-nodeID를 이용하여 Global network에서 global-nodeID를 관리하는 노드를 찾는다. 그리고 자신의 subnet에서 local-nodeID를 관리하는 노드에게 데이터를 복사한다. Global network은 topology를 고려하지 않은 Chord 이다. Subnet에서 데이터를 복사하여 자신의 subnet에 존재하는 다른 노드들이 같은 데이터를 찾을 때 자신의 subnet에서 데이터를 찾을 수 있다.

4. 결 론

현재 잘 알려진 P2P 시스템은 topology를 고려하지 않으므로 데이터를 삽입하거나 탐색할 때 물리적인 latency가 매우 클 수 있다. 그것은 데이터를 lookup할 때 성능 저하의 요인이 될 수 있다. 기존의 Topology를 고려한 시스템들은 계층적 구조를 형성하므로 super-network leader들의 부담이 매우 크며 leader들이 fail 되었을 때 leader의 교체 비용이 크다. 우리가 제안하는 시스템은 모든 노드들은 Global network상에 존재하기 때문에 leader에게 메시지를 보낼 필요가 없이 데이터를 찾을 수 있다. 또한 자신의 subnet에 데이터를 복사하여 subnet에 있는 다른 노드가 자신의 subnet에서 데이터의 찾는 확률을 높여서 lookup의 효율성을 향상시킨다. 향후 연구 과제는 데이터의 복사에 의한 저장장소의 낭비를 보다 효율적으로 줄이기 위한 replication의 정책과 fault-tolerance에 대한 연구를 할 것이다.

참 고 문 헌

[1] Napster. http://www.napster.com

[2] Gnutella. http://www.gnutella.com

[3] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A Scalable Content-Addressable Network," In Proceedings of SIGCOMM (August 2001), ACM.

[4] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek and H.Balakrkshnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," In Proceedings of SIGCOMM (August 2001)

[5] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peerto-peer Systems," In Proceedings of IFIP/ACM Middleware 2001 (November 2001).

[6] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing," Tech. Rep. UCB/CSD-01-1141, UC Berkeley, EECS,2001.

[7]K. Shin, S. Lee, G.. Lim, H. Yoon, and J. S. Ma,
" Grapes: Topology-based Hierarchical Virtual Network for Peer-to-peer Lookup Services," In Proceedings of the International Conference on Parallel Processing Workshops (ICPPW' 02), 2002.