

P2P 네트워크에서 가용성 향상 기법

김희정^o, 손영성

한국전자통신연구원, 컴퓨터 소프트웨어 연구소

Availability Improvement Strategy in Peer-to-Peer Networks

Hee-Jeong Kim^o and Young-Sung Son

Computer Software Research Laboratory, ETRI

요 약

본 논문은 P2P 네트워크에서 어플리케이션 레벨의 연속성있는 저장 시스템에 대해서 살펴본다. 인터넷 클라이언트의 성능 향상과 인터넷 접속 기술의 발달로 Peer-to-Peer (P2P)라는 키워드로 대표되는 다양한 파일 저장, 검색, 공유 어플리케이션이 소개되었다. 그러나, 기존의 서버 기반의 분산 저장 방식은 xDSL 기반의 가변 IP 기반의 현재 인터넷 환경에서 불안정한 성능을 보이고, 파일을 저장하고 나서 검색 발견의 확률이 떨어지는 문제점을 가진다. 본 논문에서는 시스템의 가용성(Availability)을 높이기 위한 파일의 분할 저장, 중복 저장 방법을 소개하고 이의 효과를 분석해 보인다.

1. 서론

인터넷과 PC의 발달은 정보의 분산과 공유를 가속화하였으며, 기존 클라이언트-서버 컴퓨팅 환경은 분산 다중 서버 환경으로 바뀌었다. 또한 정보의 원천이 웹과 데이터베이스에서 개인 PC까지 다양해졌고, 검색 포털의 한계, 즉 정보 탐색 범위 제한, 부정확성, 개인들의 검색에 대한 욕구 불만 등의 문제로 사람들은 다른 대안을 찾기 시작했다. 사용자 스스로 컴퓨터 자원과 서비스를 개인의 컴퓨터 사이에 서로 공유하는 P2P (Peer-to-Peer) 컴퓨팅이 등장하였다 [1, 2, 3].

P2P 컴퓨팅은 인터넷에서 저장 공간, CPU 사이클, 콘텐츠, 사람과 같은 자원을 활용하는 응용 분야이다. 분산된 자원에 접근하는 것은 IP가 고정적이지 않고 접속이 불안정한 환경에서 시스템이 작동하는 것을 의미하며, 각 피어는 충분히 혹은 완전히 분산되어 자율성을 가져야 한다. 최근

에는 P2P 컴퓨팅 분야에서 중복 저장(redundant storage), 저장의 영구성(permanence), 인접 피어 선택(selection of nearby servers), 익명성(anonymity), 검색(search), 인증(authentication), 명명(naming) 등과 같은 연구가 진행 중이다. 하지만, 이런 여러 가지 특징에도 불구하고, 대부분의 P2P 시스템에서 가장 중요한 기능은 효율적으로 데이터를 위치시키고(location) 탐색하는 것이다 [3, 4, 5, 6].

초기 P2P 시스템에서는 Napster에서와 같이 자원의 목록과 그 위치를 가지고 있는 서버를 두거나, Gnutella와 같이 찾는 자원에 대한 질의를 자신이 알고 있는 피어들에게 방송함으로써 자원에 대한 탐색을 실행하였다. Napster [1]와 Gnutella [2]의 방식은 확장성 문제를 안고 있다. 그래서 확장성의 문제를 극복하기 위해서 CAN, Chord, Pastry와 같은 구조화된 시스템을 발표하였다. 이러한 구조화된 시스템은 분산 해시 테이블(Dynamic Hash Table, DHT)을 사용하여 확장성과 라우팅 성능을 향상하였다.

이러한 다양한 연구결과가 현실적인 분야에 적

용되는데 가장 큰 문제점은 시스템을 구성하는 각 노드의 영속성 부재이다. 각 노드는 P2P 시스템 전용이 아니라 필요에 따라 시스템에 참여하기 때문에 전체적인 안정된 성능을 기대하기가 어렵다. 본 논문에서는 이러한 문제점을 극복하기 위해서 중복 저장(할당), 분할 저장(할당) 기법을 소개하고 이의 효과를 분석해 본다.

2. P2P 네트워크

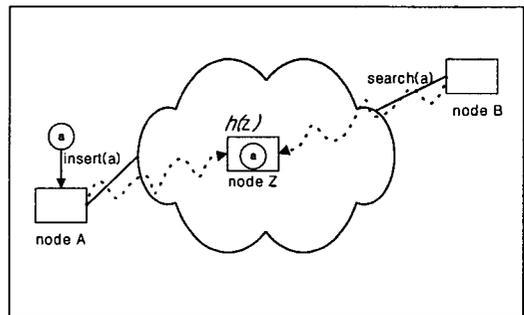
2.1 P2P 기술

P2P 기술은 불특정 다수가 참여하는 분산시스템이며 검색의 효율성, 익명성, 저장 비용의 분산 등의 특징을 가진다. P2P 구조에서는 각 클라이언트의 네트워크 상황도 고정적이지 아니며 PC의 작업 로드 상황도 안정적이지 않다. 특히 PC의 도메인도 없고 IP가 고정되어 있지 않는 것이 일반적이기 때문에 매 서비스시에 새로운 네트워크 상황을 초기화하는 작업 등이 필요하다. 또한, 사용자의 고의 또는 실수로 Peer가 다운되는 경우를 고려해야하는 어려움이 있다. Napster [1]의 경우 중앙 서버를 두어 Napster 사용자 모임 안에서 파일을 얻을 수 있도록 인덱스를 저장한다. 사용자는 파일을 얻기 위해서 중앙 서버에 먼저 질의를 하여 파일의 이름과 그 파일을 얻을 수 있는 IP 주소를 받아오게 된다. Gnutella [2]의 경우에는 플러딩(flooding)을 이용해서 파일을 찾는데 플러딩의 양이 어떤 시점에서부터 감소하기 때문에 실제로 시스템에서 파일을 못 찾을 수도 있다.

2.2 P2P 파일 저장 방법

기존의 P2P 파일 저장 방법 [4, 5, 6]은 다음과 같다. 각 노드는 전체 시스템의 유일한 해쉬함수($h(x)$)를 가진다. 이 해쉬함수는 자신의 대표 식별자를 만들기 위해서 사용되고 파일 저장, 검색 시에도 사용된다. 일반적인 P2P 파일 저장 방법은

그림 1과 같다. 파일을 저장하기 위해서는 노드A는 저장할 파일(a)의 특성(파일 이름, 크기, 생성날짜, 그 외 정보)과 해쉬함수($h(x)$)를 이용해서 저장 위치 키(destination key)를 결정한다. 이 저장위치키를 이용해서 이 값을 담당하는 노드 Z를 찾아낸다. 그리고 파일(a)을 노드 Z에 저장한다. 노드 B에서 해당 파일(a)를 찾기 위해서는 해쉬함수($h(x)$)를 이용해 알아낸 저장위치키를 이용해서 노드Z를 찾아서 파일(a)를 얻는다.



[그림 1] P2P 파일 저장 방법

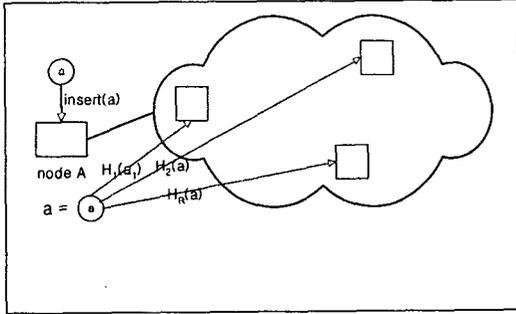
이 방식은 노드 Z가 P2P 네트워크에서 탈퇴하면 저장했던 파일(a)는 함께 접근 불가능해진다. 이러한 문제를 해결하기 위해서 가용성 향상 기법이 필요하다.

3. 가용성 향상 기법

3.1 복제(Replication) 방법

복제 방법은 저장할 파일을 P2P 네트워크의 여러 노드에 복제해서 중복 저장한다. 복제하는 방법으로 다중 해쉬 방식과 재귀 해쉬 방식이 있다. 다중 해쉬 방식은 복제할 수(R개)만큼 해쉬 함수($h_1(x)$, $h_2(x)$, $h_3(x)$, ..., $h_R(x)$)를 만들어 복제를 저장할 때 사용한다. 재귀 해쉬 방식은 해쉬함수의 결과값을 다시 해쉬한다. 맨 처음 파일 저장시의 해쉬 함수는 $h(x)$ 이고 두번째 복제 파일 저장시의 해쉬 함수는 $h(h(x))$ 가 된다. 두 방식은 성능면에서 차이는 없다. 본 논

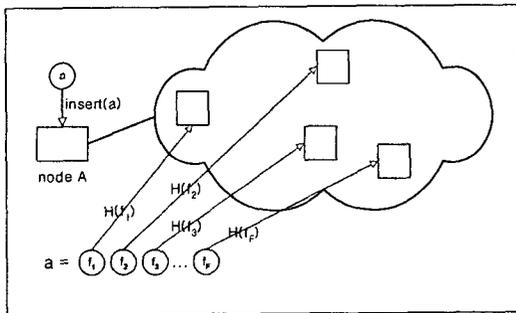
문에서 다중 해쉬 방식을 채택하였다. 다만, 복제된 파일이 같은 노드에 저장되는 것은 금지한다. 복제 방식은 복제하는 횟수만큼 가용성이 향상되는 것을 보장한다.



[그림 2] 복제 방법

3.2 분할(Fragmentation) 방법

분할 방법은 저장할 파일을 다수(F개)의 조각(f_i)으로 분할해서 각각을 P2P 네트워크 저장하는 방식이다. 이 방법은 파일을 분할하지 않고 저장할 경우 해당 노드가 탈퇴하면 전체 파일을 모두 잃어버릴 수 있기 때문에 부분적이라도 파일 사용이 필요할 때 적합하다. 그리고 모든 파일 조각이 접근 가능할 경우는 병렬 수집이 가능한 장점이 있다.

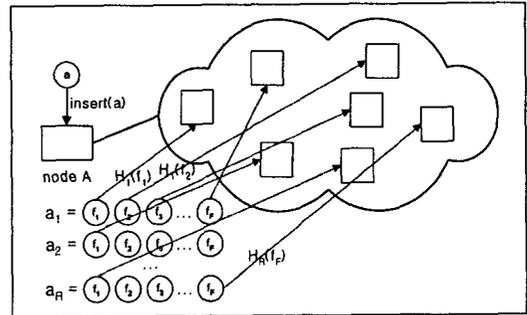


[그림 3] 분할 방법

3.3 복제 분할(Replicated Fragmentation) 방법

이 방법은 위의 복제 방법과 분할 방법을 혼합한 것으로 파일을 다수(F개)의 조각(f_i)으로 분할

한 뒤 각 조각을 다수(R개)로 복제해서 각각을 저장하는 방식이다. 그림2에서 복제 분할 방법을 설명하고 있다. 파일(a)는 R개 복제되고 각 복제파일(a_i)는 F개의 조각(f₁, f₂, f₃, ..., f_F)으로 분할되었다. 이 각각의 조각을 해쉬함수 H_i(x)를 이용해서 저장할 노드를 정하도록 하였다.



[그림 4] 복제 분할 방법

4. 가용성 분석

본 절에서는 앞에서 소개한 가용성 향상 기법의 효과를 비교해 본다.

분석에 필요한 파라미터는 다음과 같다.

- 네트워크에 참여하는 노드 수 : N
- 결함(Failure)있는 노드 수 : M
- 파일 조각의 수 : F
- 복제 파일의 수 : R

4.1 No replication, no fragmentation

파일 복제나 파일 분할을 사용하지 않았을 경우에는 파일이 가용하지 않을 확률은 파일을 저장하고 있는 노드가 결함이 될 확률이다.

$$Failure(a) = \frac{M}{N}$$

4.2 No fragmentation, replication

파일 복제만 적용한 경우에는 복제 파일이 저장

된 모든 노드가 결함이 될 확률이다. 단, 복제 파일은 같은 노드에 저장될 수 없으므로 복제 파일은 모두 다른 노드에 저장되었을 경우를 고려해야 한다.

$$Failure(a) = \frac{M}{N} \times \frac{M-1}{N-1} \times \dots \times \frac{M-R+1}{N-R+1}$$

4.3 Fragmentation, no replication

파일 분할만 적용한 경우에는 모든 조각 파일이 접근가능해야 한다. 단 하나의 조각 파일이라도 저장된 노드가 결함이 있다면 전체적으로 파일 재구성에 결함이 발생한다. 조각 파일은 복제 파일은 같은 노드에 저장될 수 있다.

$$Failure(a) = 1 - \left(\frac{N-M}{N} \right)^F$$

4.4 Fragmentation, replication

파일 분할과 파일 복제가 함께 적용한 경우에는 조각파일의 가용 확률을 먼저 계산해야 한다. 조각 파일의 가용 확률은 조각파일이 저장된 모든 노드가 결함이 될 확률로부터 구할 수 있다.

$$Available(a_i) = 1 - \frac{M}{N} \times \frac{M-1}{N-1} \times \dots \times \frac{M-R+1}{N-R+1}$$

이제 전체 파일의 가용 확률은 개별 조각 파일의 가용 확률의 곱이다.

$$Available(a) = \left(1 - \frac{M}{N} \times \frac{M-1}{N-1} \times \dots \times \frac{M-R+1}{N-R+1} \right)^F$$

따라서, 파일 분할과 파일 복제가 함께 적용된 경우의 파일이 가용하지 않을 확률은 다음과 같다.

$$Failure(a) = 1 - \left(1 - \frac{M}{N} \times \frac{M-1}{N-1} \times \dots \times \frac{M-R+1}{N-R+1} \right)^F$$

5. 결론

본 논문에서는 인터넷 노드의 능동적인 참여로 구성되는 P2P 네트워크에서 가용성을 향상시키기 위한 방법을 소개하였다. 해쉬함수를 이용하여 객체의 배치 및 검색을 수행하는 P2P 네트워크에서 객체의 접근 가능성이 매우 중요한 요소이다. 이러한 객체에 대한 접근 가능성을 가용성 분석을 통해서 복제 방법, 분할 방법, 복제와 분할 방법을 동시에 사용한 경우에 대해서 살펴보았다.

추후 연구분야로는 본 논문에서 분석한 가용 모델을 확장하여 복제 방법과 분할 방법이 P2P 네트워크의 데이터 검색에 미치는 영향에 대해서 수행할 계획이다.

[참고문헌]

- [1] Napster. <http://www.napster.com>
- [2] Gnutella. <http://gnutella.wego.com/>
- [3] I. Clark, O. Sandberg, B. Wiley and T.W. Hong, "Freenet: A distributed anonymous information storage and retrieval system in designing privacy enhancing technologies," International Workshop on Design Issues in Anonymity and Unobservability, LNCS 2009, 2001.
- [4] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems," 18th IFIP/ACM International Conference on Distributed Systems Platforms, 2001.
- [5] I. Stocia, R. Morris, D. Karger, M. Frans Kaashoek, H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," SIGCOMM, 2001.
- [6] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, "A Scalable Content-Addressable Network," SIGCOMM, 2001.
- [7] William Pugh, "Skip Lists: A Probabilistic Alternative to Balanced Trees," Communication of ACM, June 1990.