Attenuated Bloom Filter의 전달을 이용한 APS의 성능 개선

권남° 박재현
중앙대학교 컴퓨터공학과
kwonnam@hsc.cau.ac.kr°, hyunie@cau.ac.kr

Efficiency Improvement of APS using the delivery of Attenuated Bloom Filter

Nam Kwon°, Jae-Hyun Park
Dept. of Computer Science and Engineering, Chung-Ang university

요 약

P2P 검색 네트워크에서 노드간의 정확한 자료를 검색하는 방법은 중요한 연구주제이며 다양한 종류의 기법들이 제안되어 있다. 본 논문은 그 중에서 Gnutella 네트워크를 기반으로 기존의 검색 결과를 이용하여 다음 번 자료 검색 시 높은 확률로 정확한 자료를 찾을 수 있게 도와주는 검색 방법인 APS 방식의 단점을 지적하고 거리에 따른 자료의 분포를 알 수 있는 인덱스인 Attenuated bloom filter를 교환하고 이를 검색에 이용하여 APS의 문제점들을 해결하고 동시에 전반적인 검색성공률을 높이는 방법을 제안한다.

1. 서론

Peer to Peer(이하 P2P) 검색 네트워크는 노드간의 직접연결을 통하여 다양하게 자료를 검색할 수 있다. 따라서 검색하는 방법에 따라 다양한 종류의 알고리즘들이 존재한다. Gnutella[1]의 경우 자료를 검색하는 방법에 따라 아무런 기존의 정보 없이 검색을 수행하는 blind search와 meta data를 이용하여 검색에 이용하는 informed search로 크게 분류할 수 있는데 본 논문에서는 informed search 기법 중 Random k-walker기법을 기반으로 검색성공률을 높이는 방식을 제안한 Adaptive Probabilistic Search[2](이하 APS)를 살펴보고 APS가 수동적으로 index 구성하므로 발생하는 검색성공률의 저하를 지적한다. 이를 해결하기 위해 연결된 노드들에게 거리에 따른 자료의 정보를 표현할 수 있는 attenuated bloom filter[3]를 전달하여 검색할 때 이용하도록 한다. 이를 통해 APS의 index를 능동적으로 구축함으로 APS 검색 방식의 단점을 개선하고 APS의 전반적인 검색 성능이 높이는 방법을 본문에서 제시한다.

2. 관련연구

Gnutella는 검색어를 모든 연결에 flooding 하는 방식인 BFS방식을 사용한다. BFS는 TTL만큼의 홉 안에 있는 모든 노드의 자료를 검색할 수 있는 장점을 가진다. 하지만 많은 양의 중복된 query를 발생시킴으로써 쓸모없는 패킷을 발생시켜 부하를 일으키는 문제점이 발생하게 된다.

이에 따라 모든 연결된 노드에 query를 전송하지 않고 random 하게 k개의 연결만 선택하여 전송하는 방식인 random k-walker 방식이 고안 되었다.

Random k-walker[4] 방식은 중복된 메시지의 발생을 줄여 대역폭 낭비를 줄였다. 하지만 모든 노드에게 검색 메시지를 전달할 수 없어서 검색 성능이 떨어지는 단점이 있다.

그래서 Random k-walker를 기반으로 하여 검색할 때 부하를 줄이고 검색 성공률을 높이는 다양한 방식이 고안 되었다. 대표적인 방식이 기존의 검색 결과를 이용하여 다음 검색에 이용하는 informed 방식인 APS가 있다.

APS는 keyword 별로 연결에 대한 검색성공 비율을 index로 저장한다. 검색을 반복하여 각 query 대한 성공여부를 다음 검색 시에 이를 이용해서 확률에 따라 k-walker를 전송하는 방법이다.

APS는 별도의 메시지의 발생 없이 index를 구성하고 유지할 수 있어서 구성이 간단하며 구현이 쉽고 일반적으로 자주 검색되어지는 keyword에 대해서 높은 검색 성공률을 보장 할 수 있다. 또한 인덱스의 값에 따른 우선순위로 전송하지 않고 단지 전송될 확률만 보장하게 되므로 새로 참여한 노드에게도 query가 전송되어질 수 있는 기회가 주어진다.

하지만 keyword 별로 index를 유지하기 때문에 P2P 네트워크상에 처음으로 등장하는 keyword나 자주 검색되어지지 않는 keyword의 경우는 검색을 하기위한 keyword별 index가 존재하지 않거나 네트워크 전체에 걸쳐 소수만 존재하기 때문에 검색 성공률이 Random k-Walker와 비슷해진다. 또한 peer의 참여와 탈퇴가 빈번히 일어나는 경우 기존의 연결에 대한 index 정보를 신뢰 할 수 없기 때문에 검색 성능이 낮아진다.

다음으로 본 논문에서 사용할 Attenuated Bloom Filter는 d개의 bloom filter들로 이루어진 배열로 표현된다. 배열의 첫 번째 bloom filter를 depth 1로 하고 i번째 bloom filter는 depth i로 표현하여 i번째 홉에 해당 하는 거리에 있는 노드들의 자료를 표현 할 수 있는 장점이 있지만 서로 다른 연결에서 다중 홉 거리에 있는 같은 노드의 자료가 중복될 수 있고, depth가 증가할수록 거의 모든 bit가 1로 저장되어 있어 정확한 검색이 어렵다. Attenuated Bloom filter는 DHT방식인 tapestry에 적용[3]되어 사용된다. 검색하는 자료의 전체이름을 해싱하여 Attenuated bloom filter를 통해 자료를 검색하여 가장 낮은 depth 즉, 가장 가까운 거리에 자료가 존재하는 노드로 query를 전달하는 greedy 방법을 취한다. 비록 DHT에서 사용하지만 Gnutella에서도 자료를 검색할 때 자료의 이름을 keyword로 나누어 검색하는 방법[5]을 사용하기에 동일하게 적용할 수 있다.

3. 개선 방안

본 논문이 제안하는 알고리즘은 attenuated bloom filter 인덱스를 한 홉 거리의 노드에게 전송하여 주변 노드에게 알리고

검색 시에 이를 이용하도록 하여 수동적으로 index를 만들지 않고, 능동적으로 index를 구성하도록 하는 것이다. 이와 같이 attenuated bloom filter 인덱스를 가지고 있어서 검색 시 이용하면 인덱스 구성시간을 줄일 수 있고 최초로 검색하는 검색어라도 attenuated bloom filter를 이용하여 높은 확률을 가지고 검색 할 수 있어 노드의 참여와 탈퇴가 빈번한 동적인 환경에서도 빠른 적응이 가능하다.

Attenuated bloom filter를 사용할 경우 위에서 언급했듯이 거리에 따른 자료의 존재 여부를 알 수 있는 장점이 있다.

이러한 특징을 이용해서 검색에 사용한다. 먼저 검색을 시작 하거나 검색 메시지를 받아서 query를 전송 하게 될 때, 전송 받은 메시지의 키워드가 APS 인덱스에 존재하지 않으면 해당 키워드에 대한 인덱스를 이웃 노드별로 초기 비율 값을 가지도록 초기화하여 생성한다. 다음으로 attenuated bloom filter 인덱스를 이용하여 검색한다. 연결마다 유지 되는 attenuated bloom filter 인덱스에서 keyword를 검색하여 각 depth 마다 and 연산을 하고, and 연산의 결과가 0이 아닐 경우 미리 정해놓은 가중치 a에 $1 / 2^{(depth)}$ 을 곱하여 해당 하는 연결 노드의 APS 비율 값에 더한다. 이와 같은 연산을 bloom filter의 depth와 TTL중에서 작은 값만큼을 반복한다. 주어진 횟수만큼 반복해도 attenuated bloom filter index에서 해당 keyword가 검색이 안 될 경우 해당 이웃 노드로는 검색어가 존재 하지 않으므로 APS index에서 검색되지 않은 연결에 대한 비율 값을 최소값으로 초기화 시켜 준다. 그 후 APS에서 사용하는 검색 방식과 동일하게 attenuated bloom filter 값이 반영된 index에 있는 비율 값에 기반 하여 확률에 따라 query를 전송한다. query를 전송한 후 bloom filter에 의하여 더해졌던 가중치는 이전 상태로 되돌려 준다.

검색 알고리즘

```
ABF is Attenuated Bloom Filter
if (keyword is not in the APS index)
      init(APSIndex(keyword));
a = initValue;
weight=0;

neighbor[] = APSindex(keyword)
i= min(TTL, Attenuated Bloom filter's depth)
for i = 0 to neighbor's length
{
      ABF = ABFIndex(neighbor[i])

      for 0 to i -1
            if ( is ABF(i,keyword) )
                  weight +=a*1/2^(depth);
      if (weight is 0)
            initProbability(neighbor's probability);
      else neighbor's probability += weight;
}
이하 APS 와 동일
```

Attenuated bloom filter를 이웃노드에게 전달하는 방법은 먼저 네트워크에 참여한 노드들은 ping의 발생 주기에 맞춰 자신의 자료 정보를 bloom filter로 요약하여 depth를 0으로 두고 자신의 가지고 있는 모든 attenuated bloom filter를 or 연산을 통하여 요약된 값을 depth 1부터 저장하되 마지막 depth 부분은 제거한다. 만들어진 인덱스 정보를 저장한 뒤 ping 메시지에 덧붙여서 모든 이웃 노드에게 전달한다. 이때 만들어진 Attenuated bloom filter가 기존에 저장된 index와 동일하다면 ping에 덧붙이지 않는다. ping을 통해 전송된 attenuated bloom filter는 그대로 메시지를 전송한 노드의 인덱스 정보로 저장한다. 마지막으로 ping에서 attenuated bloom filter부분을 제거한다.

Attenuated Bloom Filter 전달 알고리즘

```
ABF is Attenuated Bloom Filter
sendPing()
{
    made Ping message;
    ABF == Attenuated Bloom Filter
    ABF =  initiation Atteunuated Bloom filter ;
    ABF's depth 0  -> set(my data)
    tempABF = or(all in Attenuated Bloom filter
index)

    for i=1 to depth-1
        ABF's depth i = tempABF's depth i-1;

    if(compare( ABF, beforeABF) is false)
    {
        beforeABF = ABF;
        add(ping, ABF);
    }
    send(ping);
}
receivePing(ping)
{
    if( there is another message in ping)
    {
        ABF = another message in ping;
        set ABF index by node that send ping;
        remove another message from ping;
    }
    send(ping)
}
```

4. 실험

앞에서 언급한 알고리즘을 P2P 네트워크에 적용한 결과를 확인하기 모의실험기[6]를 이용하였다. 실험은 서로 다른 random seed를 이용하여 3번 반복한 결과의 평균을 이용하였다. 실험에서 사용한 전체 노드의 개수는 1000개를 사용했으며 노드의 개수를 고려하여 TTL은 4로 하였다. 네트워크의 연결은 power law[7] 형태의 graph을 이용하였고 일정 주기마다 random하게 노드가 참여 또는 탈퇴 하는 동적인 환경을 구축했다. 전체노드의 약 80% 노드는 네트워크상에 존재 하며, 전체 노드 중 25%[8]중 항상 네트워크에 참여한 상태로 있게 하였다. 또한 각 노드는 gnutella 네트워크의 평균인 100[9]개의 자료를 가지고 있으며 자료의 분포는 Zipf 분포[7]를 사용하였다. 검색어는 서로 다른 20개의 검색어를 균일한 분포로 각각 다른 노드에서 검색을 시작 하였다. 이 논문에서 사용하는 bloom filter의 크기는 2046bit로 하였다.

실험에 사용한 주요 인자들	
노드의 개수	1000
keyword의 총 개수	20000
노드당 document 개수	100
검색 횟수	20000
walker의 개수	1~6
Attenuated Bloom filter의 크기	1KB(2046bit * 4)
TTL	4
Network의 연결	Power Law
자료의 분포	Zipf distribution

표 1 실험에 사용한 주요 인자

4.1 검색 성공률

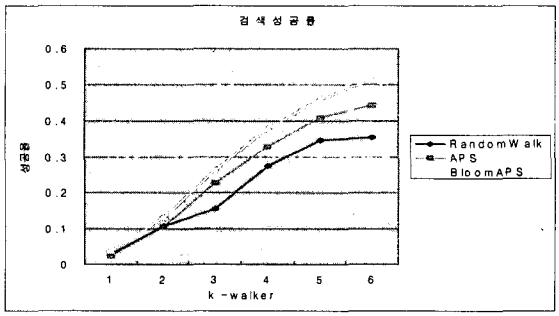

그림 1 k walker 에 따른 검색 성공률

그림 1에서의 그래프를 보듯이 k-walker에서 k가 6일 때 Attenuated bloom filter를 적용한 경우 검색 성공률이 약 51.3%, APS가 44.2%, Random k-Walker의 경우 35.3%로서 본 논문에서 제안한 방식이 기존의 APS 경우 보다는 약 15% 정도 Random k-walker보다는 약 45%정도 높은 성공률을 보이는 것을 알 수 있다. 실험 결과에서 알 수 있듯이 주변노드의 정보를 알고 이웃노드에게 검색메시지를 전달한 경우가 높은 성공률을 보임을 알 수 있다.

4.2 검색 진행에 따른 성공률의 변화

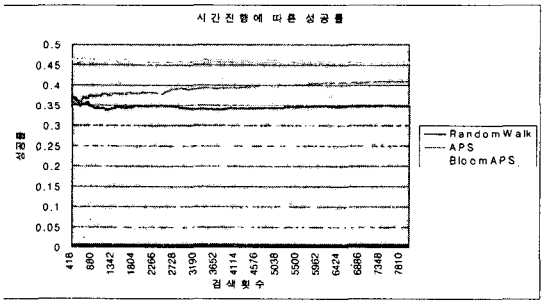

그림 2 k=5 일때 시간진행에 따른 검색 성공률

그림 2의 그래프는 검색의 반복에 따른 누적 검색 성공률을 검색의 반복 횟수로 나눈 그래프로서 검색 반복에 따른 검색 성공률의 변화를 알 수 있는 그래프로 Walker가 5일 때의 모습이다. APS의 경우 예전에 검색했던 결과를 이용해서 검색에 이용하므로 검색을 시작하는 초반에는 검색 성공률이 높지 않다가 점차 증가 하는 모양을 보인다. 그래프를 보면 APS의 경우 검색을 시작할 때의 최소값 34.9%에서 검색을 반복함에 따라 차츰 증가하여 평균 최대값 41.0% 까지 증가 한다. 반면 attenuated bloom filter를 적용한 경우 검색 성공률의 변화가 거의 없이 꾸준히 높은 성공률을 보임을 확인할 수 있다.

5. 결론과 향후 계획

본 논문에서는 P2P 네트워크에 참여한 한 홉 거리의 노드끼리 attenuated bloom filter를 교환함으로써 이웃 노드의 정보를 알게 되었고, 이를 이용하여 검색 메시지를 보낼 때 기존 APS보다 높은 확률로 전송할 수 있게 되어 APS에 비해 약 15%정도의 전체 적인 성능향상이 있었고 APS의 경우 검색 성공률이 안정화되기까지 각각의 검색어 마다 약 250번의 검색을 수행했으나 attenuated bloom filter를 사용한 경우 초반부터 일정한 성공률을 보여 앞서 지적했던 단점을 해결했음을 알 수 있다.

하지만 실험한 노드들의 개수와 TTL을 정확히 반영하지 못하여 실험에 부족한 부분이 있었고 bloom filter의 크기에 따른 검색 성공률의 비교가 필요하다고 생각된다.

앞으로 Attenuated Bloom filter를 적용할 경우 발생하는 추가적인 대역폭 소비와 중복된 자료들을 구분하지 못하는 문제점에 대한 연구를 진행할 예정이며 노드의 개수와 TTL을 증가시켜 실험할 계획이다.

6. 관련자료

[1] http://rfc-gnutella.sourceforge.net/

[2] D. Tsoumakos, N. Roussopoulos, "Adaptive probabilistic search in peer-to-peer networks", *Proc, of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, 2003

[3] Sean C. Rhea, John Kubiatowicz, "Probabilistic Location and Routing", *Proc, of IEEE INFOCOM'2002*, 2002.

[4] Rohrs, C. "Query Routing for the Gnutella Network", *Lime Wire*, may 2002

[5] Qin Lv, Pei Cao, Edith Cohen, Kai Li, Scott Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks", Proc. 16th International Conference Supercomputing(ICS'02), ACM Press, 2002

[6] Joseph S. "NeuroGrid: Semantically Routing Queries in Peer-to-Peer Networks." *In Proceedings of the International Workshop on Peer-to-Peer Computing (co-located with Networking 2002)*, 2002

[7] D.Tsoumakos and N. Roussopoulos. "A Comparison of Peer-to-Peer Search Methods", *In WebDB*, 2003.

[8] Matei Ripeanu, Ian Foster, and Adriana Iamnitchi, "Mapping the Gnutella network: Properties of large-scale peer-to-peer systems and implications for system design," *Internet Computing, vol. 6, no. 1, pp. 50-57*, 2002.

[9] S. Saroiu, P. K. Gummadi, and S. D. Gribble. "A measurement study of peer-to-peer file sharing systems", *In Proceedings of Multimedia Computing and Networking*, January 2002