IP주소를 이용한 지역위치정보검색 시스템

조민정° 변성원
KT
{mini°, swbyon}@kt.co.kr

Location Information Retrieval System using IP Address

Minjeung Cho°, Sungwon Byon
Korea Telecom

요 약

본 논문은 IP주소로 파악한 사용자의 지역정보를 이용하여 응용서비스를 제공하는 시스템에 관한 것으로, 대량의 검색 질의를 하는 포털사이트에 최적화된 서비스를 하기 위하여 검색속도를 향상시킬 수 있는 데이터 구조를 적용한다. IP주소를 이용한 지역위치정보 검색시스템은 array를 이용한 flag, hashing을 이용한 cache, trie, 메모리 database 등의 자료구조를 이용하며, 각각을 실험을 통해 검색성능 향상에 미치는 영향을 평가한다.

1. 서 론

지역정보 서비스가 인터넷 포탈사이트 업계의 화두가 된 이후로 몇 년이 지났지만 뚜렷하게 활성화되는 모습이나 수익성을 보이지 못하고 있다. 지역정보 서비스 활성화에 가장 큰 걸림돌 중 하나는 사용자의 관심지역을 알아내야 한다는 것이다. 사용자가 관심지역정보를 미리 세팅해놓거나 로그인하기 전에는 사용자의 지역정보를 알아내기 어렵고, 사용자가 검색시 지역정보를 입력하도록 교육을 시키려는 시도도 있었지만 성공하지 못했다. 사용자의 지역위치정보를 자동으로 알아내기 위한 방법 가운데 사용자가 접속한 IP 주소를 이용하여 위치정보를 추정하는 방법이 있다. 2장에서는 IP주소를 이용한 지역위치정보 시스템 구축, 3장에서는 IP주소를 이용한 위치정보 검색, 4장에서는 검색실험 결과에 대해서 논하고 5장에서 결론으로 향후 전망에 대해 기술한다.

2. IP주소를 이용한 위치정보 검색 시스템 구축

IP주소로 위치를 추정하는 방법은 크게 고정 IP 주소인 경우와 유동 IP 주소인 경우로 나뉜다. 고정IP주소의 위치정보는 IP할당내역을 관리하는 서버로부터 주기적으로 bqch작업으로 가져오고, 유동IP주소의 위치정보는 사용자가 인터넷에 접속하여 IP를 할당받으면 실시간으로 수집하여 가입자정보DB의 주소필드와 join하여 위치정보를 얻는다.

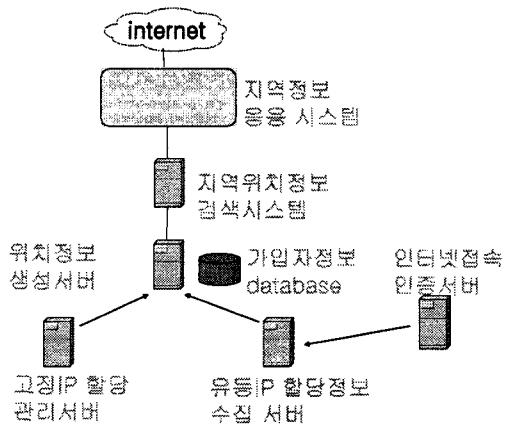
지역위치정보 검색 시스템은 IP로 지역위치정보를 검색하여 응용 시스템에 지역코드값을 넘겨주고, 메모리 cache에 저장하여 동일 IP에 대한 질의가 올 경우에 DB에 접속하지 않고 처리한다.

지역코드는 8byte로 이루어져있으며 각각의 2-4byte들은 광역시/도, 시군구, 읍면동을 계층적으로 나타내며 전체 지역코드의 숫자는 약 3400여개이다.

유동IP주소는 할당된 IP주소 각각의 위치가 다르므로 하나하나씩 DB에 저장되지만, 동일한 위치로 할당된 인접한 고정IP주소들은 크기를 줄이기 위해 시작IP주소와 끝 IP주소의 2개 IP로 나타내어지는 하나의 IP블록으로 통합된다. 즉, 147.6.22.0 부터 147.6.22.255 까지가 모두 우면동으로 할당되었다면, 데이터베이스에는 하나의 row로 저장된다.

인터넷 포털사이트 등 응용시스템은 하루에 1억페이지뷰까지 검색을 요청할 수도 있으므로 검색속도가 매우 빨라야 하고, 피크타임에는 초당 수천건의 검색query를 처리할 수 있어야 한다.

IP주소를 이용한 검색성능 향상에 관해서는 라우터와 스위치에서 사용되는 알고리즘에서 많이 논의가 되었다. 그러나, 라우터는 24포트 중에 한곳으로 매핑하면 되지만 지역정보검색은 3400개의 지역코드로 매핑시켜야 하고, IP블럭의 수도 라우터는 5만개정도이지만 지역정보검색은 80만개 이상이므로 복잡도가 높다. 또한 수행되는 환경과 연산이 수행되는 하드웨어의 가격과 구성도 다르다.



[그림 1] IP주소를 이용한 지역위치정보생성 및 검색 시스템

3. 검색 성능 향상을 위한 자료구조

3.1 검색 수행 판단을 위한 flag

KT는 인터넷 접속시장의 53% 정도만을 점유하고 있으므로 지역위치정보를 요청하는 전체 IP주소의 53%만이 데이터베이스 검색을 통해서 응답을 얻을수 있고 나머지는 데이터베이스에 없으므로 검색해도 정보를 얻지 못한다. 그러므로 검색을 수행하기 이전에 미리 데이터베이스에 검색할 IP주소의 위치정보가 존재하는지를 알수 있다면 47%에 해당하는 검색 횟수를 줄일수 있을 것이다. C클래스 단위로 IP주소가 KT의 주소영역에 해당하는지를 미리 flag array로 만들어 메모리에 올려놓고 flag값에 따라 검색을 수행할지 널값을 리턴할지 결정한다. 전체 C클래스 숫자는 2^{24} 개이므로 flag를 character 로 구현할 경우에는 16M의 메모리에, bit array로 구현할 경우에는 2M의 메모리에 구현할 수 있다. 각 IP주소의 flag 값을 얻기 위한 flag array의 포인터값은 아래와 같이 계산할 수 있다.

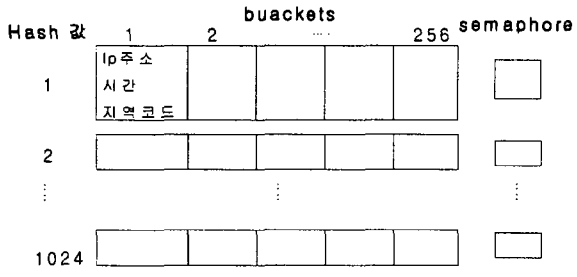
array포인터값 = (A클래스*256+B클래스)*256+C클래스

즉, 147.6.22.26의 포인터값은 (147*256+6)*256+22 = 979990 이므로 flag array의 979990번째 flag 값을 조사하면 147.6.22.26이 KT의 IP대역인지 검색을 수행해야 하는지 알 수 있다.

3.2 hashing을 이용한 cache

검색이 이루어진 후에 IP주소와 위치정보는 cache 에 저장되어 일정시간내에 동일한 질의가 오면 DB접속을 하지 않고 cache에 저장된 값을 읽어서 응답한다.

cache는 검색을 수행하지 않고 빠르게 값을 찾아내는 것이 목적이므로 hashing 함수를 사용한다. 그러나 hash 값을 넓게 분포시키면 메모리를 낭비하게 되고 공간이 너무 커져서 메모리에 다 올리지도 못하게 된다. 또 hash 값을 좁게 분포시키면 충돌의 문제가 발생하므로 bucket들을 여러개 두거나 적절히 공간을 refresh 시키는 것이 필요하다. 본 논문에서는 hash값은 1024개, bucket은 256개로 하여 2^{18} 개의 IP주소 정보를 저장하도록 하였다. 최초 DB검색하여 cache에 올려서 3분동안만 bucket에 저장하고 3분이 지난 정보는 refresh 시킨다. 256개의 bucket은 IP주소로 정렬되어있어 binary search로 비교검색 평균4번($256=2^8$ 이므로 최대 8번 이내)만에 빠르게 찾을수 있다.

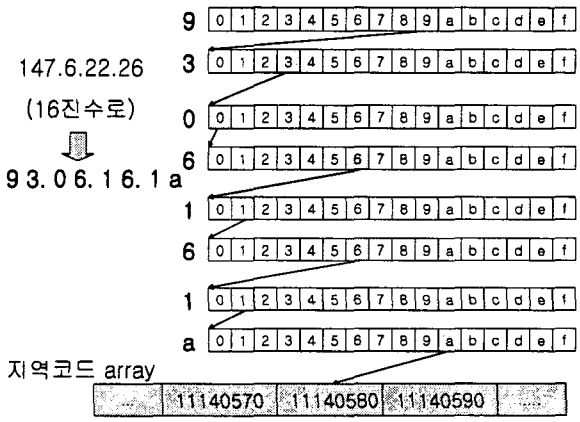


[그림2] cache 메모리 구조

cache는 빠르게 업데이트 되고, IP주소로 정렬되어 있어야 하므로, 데이터를 수정하는 동안에는 lock을 걸어 보호해야 한다. cache전체를 lock을 걸면 속도가 느려지고 각각의 bucket 하나하나마다 lock을 걸면 locking process를 처리하는 시간이 검색시간보다 길어질 수 있으므로, 정렬상태를 유지하여야 하는 hashing 값이 같은 256개의 bucket들마다 semaphore를 하나씩 공유한다.

3.3 고정 IP블럭을 저장하는 trie

유동 IP주소의 위치정보는 수시로 바뀌고 업데이트되므로 데이터베이스에 저장되지만 고정IP주소의 위치정보는 batch작업으로 1일 또는 1주일에 한번씩 업데이트된다. 그러므로 고정IP주소는 shared memory에 올려서 데이터베이스 접속을 하지 않고 바로 검색하여 응답을 줄 수 있도록 한다. 고정IP블럭을 저장하는 메모리저장구조는 검색속도를 위해 표현범위를 제한하는 cache나 flag와 달리, 빠른 검색속도와 더불어 모든 데이터범위를 정확히 빠짐없이 표현하고 있어야 한다.



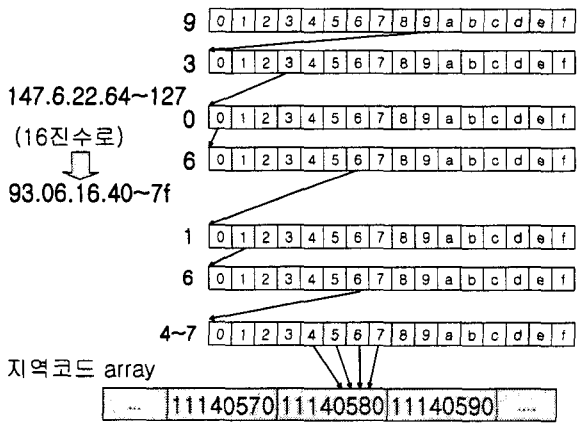
[그림3] 일반적인 IP주소를 나타내는 trie 메모리 구조

trie는 빠르긴 하지만 메모리를 많이 사용하므로 널리 쓰이지는 않고 있다. 그러나 응용프로그램의 목적에 맞게 변형된 trie들은 종종 사용되는데, 본 논문에서도 일반적인 trie를 그대로 적용하지 않고, 메모리 사용량을 줄이기 위하여 trie의 노드를 줄여 변형시켰다.

Ip블럭을 trie로 나타내기 위해서 하나의 노드는 0부터 f(10진수로 15)의 배열로 구성하고, IP주소는 16진수 8자리로 나타내고 각각의 16진수를 key값으로 8개의 노드를 따라가서 지역코드값을 얻는다. 예를들어, 147.6.22.26은 16진수로 바꾸면 93.06.16.1a이고 노드의 0부터 f까지의 array에서 숫자에 해당하는 배열값 9를 찾아서 다음 노드의 포인터값을 얻고, 다음 노드로 가서 배열값 3을 찾아서 다음 노드의 포인터값을 얻고, 다시 0, 6, 1, 6, 1, a를 차례로 찾아 8개의 노드를 거쳐보면 지역코드 값을 알 수 있다.

IP블럭은 시작IP주소와 끝IP주소로 나타내지고 그 구간에서는 지역코드가 같으므로, trie에서 하위노드의 IP값들의 지역코드가 모두 같다면 하위 노드 표현을 생략하도록

변형하면 노드의 수를 많이 줄일 수 있다. 예를들어, 147.6.22.64~127 까지의 블록은 16진수로 나타내면 93.06.16.40~7f 로 나타낼 수 있고, 93.06.16.4x 93.06.16.5x 93.06.16.6x 93.06.16.7x 의 4개의 노드 배열값으로 나타내어질 수 있다. 이렇게 하면 64개의 IP주소는 4개의 배열값으로 나타내어질 수 있고 8단계였던 노드는 7단계로 줄여질 수 있다.



[그림4] IP블럭을 나타내기위해 변형된 trie 메모리 구조

3.4 memory database

유동IP 주소가 할당되면 즉시 인터넷 접속 ID와 할당한 IP주소를 수집하여, 접속ID로 사용자 DB에서 사용자의 주소정보로 접속위치를 알아내어 지역코드로 변환하여 데이터베이스에 저장한다.

4. 실험결과

실험에 사용된 IP주소는 네이버의 1개 웹서버에 접속한 1일치 로그에서 추출하였으며 중복된 IP주소를 제거하지 않고 실제 지역정보 질의패턴을 반영하도록 하였다. 전체 9732490개 가운데 5195857개가 KT의 IP주소 대역이고 4546633개는 타 ISP의 주소 대역으로부터 들어온 IP주소였다. 실험은 각각의 flag, cahce, trie 구조를 사용한 경우와 그렇지 않은 경우의 처리속도를 비교해볼 수 있도록 행하여졌다. [표1]은 각각의 자료구조가 처리하여 결과값을 리턴한 IP주소질의의 수를 나타낸 것이다. flag는 타 ISP의 주소에 대해서 응답을 하고, cache는 반복된 IP주소를 처리한다.

flag	cache	trie	db	flag	cache	trie	db
o	o	o	o	4536633	497166	2781637	1917054
x	o	o	o	0	497166	2841886	6393429
o	x	o	o	4536633	0	2781637	2414220
x	x	o	o	0	0	2841886	6890604
o	o	x	o	4536633	1736830	0	3459027
x	o	x	o	0	1736661	0	7995829
o	x	x	o	4536633	0	0	5195857
x	x	x	o	0	0	0	9732490

[표1] 각각의 자료구조가 처리한 질의 수

flag	cache	trie	db	초당 질의 처리 수
o	o	o	o	30225
x	o	o	o	11135
o	x	o	o	24270
x	x	o	o	10420
o	o	x	o	16060
x	o	x	o	8515
o	x	x	o	10985
x	x	x	o	6844

[표2] 각각의 자료구조를 사용한 경우의 초당 처리 수

[표2]는 각각의 자료구조를 사용한 경우에 초당 질의 처리 숫자를 나타낸 것이다. 모든 자료구조를 사용했을 경우에 가장 많은 질의를 처리하고 자료구조를 하나도 사용하지 않았을 경우보다 4.4배 더 많은 질의를 처리하는 것으로 나타났다. 자료구조별로는 flag가 절반가량의 IP주소를 filtering하므로 검색성능 향상에 가장 많은 영향을 미치는 것으로 나타났고, cache보다 trie가 성능을 향상시키는 이유는 역시 trie가 처리한 Ip주소가 cache가 처리한것보다 많기 때문으로 추정된다.
실험에 사용된 서버는 HP 리눅스 4CPU 2G 메모리 이고 데이터베이스는 알티베이스 3.00이다.

5. 결론

IP주소를 이용한 지역위치정보 검색 시스템에 있어서 flag, cache, trie 의 성능을 비교해보았다. flag는 DB에 없는 Ip주소를 미리 filtering하는 역할을 하고, cache는 반복되는 질의를 처리하고, trie는 자주 업데이트되지 않는 고정Ip주소를 검색하는 역할을 하며, 각각의 자료구조는 검색성능 향상에 독립적으로 영향을 미치고 있는 것으로 나타났다.
저렴한 1대의 서버로 초당 3만건 이상의 응답을 해줌으로써 인터넷 서비스에서 지역위치정보 서비스의 다양한 응용이 가능해질 것으로 기대한다.

참고문헌

[1] 변성원, 김민경, "인터넷환경에서 지역타게팅 광고 시스템의 구현" , JCCI, 2006.
[2] 이화식, 조광원, "대용량 데이터베이스 솔루션 I, II", en-core 컨설팅, 1998
[3] Altibase, "Altibase Application Development SQL User's Manual", Releasee 4.3.1.0
[4] Ramez Elmasri, Shamknt B. Navathe, "Fundamentals of DATABASE Systems", The Benjamin/Cummings Publishing Company, Inc.
[5] Mary E.S. Loomis, "Data Management and File Structure", Prentice-Hall International Editions.