13장 검색어 자동완성 시스템

문제 이해 및 설계 범위 확정

- 입력하는 단어는 검색어의 첫 부분? or 중간 부분?
  - 첫 부분
- 몇 개의 자동완성 검색어 표시?
  - 5개
- 자동완성 검색어를 고르는 기준은?
  - 질의 빈도에 따라 인기 순위
- 맞춤법 검사 기능이나 자동수정 기능도 제공?
  - 아니요
- 질의는 영어인가?
  - 네. 시간이 허락하면 다국어 지원도 생각해도 좋다.
- 대문자나 특수 문자 처리도 해야 하나요?
  - 아뇨. 모든 질의는 영어 소문자
- 얼마나 많은 사용자 지원?
  - 일간 능동 사용자(DAU) 기준으로 천만(10million)명

요구사항

- 빠른 응답 속도
  - 100밀리초 이내
- 연관성
- 정렬
- 규모 확장성
- 고가용성

개략적 설계안

시스템은 두 부분으로 나뉜다.

- 데이터 수집 서비스(data gathering service)
  - 사용자가 입력한 질의를 실시간으로 수집하는 시스템이다.
  - 데이터가 많은 실시간 시스템은 바람직하지 않음.
  - 상세 설계에서 현실적인 안으로 교체
- 질의 서비스(query service)

데이터 수집 서비스

- 질의문과 사용빈도를 저장하는 빈도 테이블이 있다.

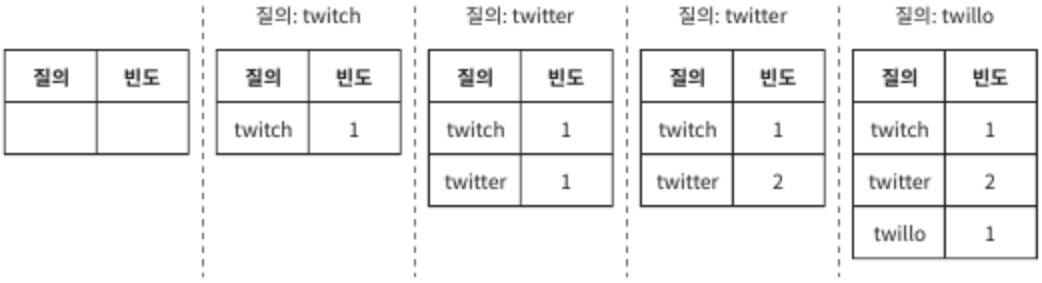

그림 13-2

질의 서비스

query	freuqency
twitter	35
twitch	29
twilight	25
twin peak	21
twitch prime	18
twitter search	14
twillo	10
twin peak sf	8

표 13-1

이 상태에서 “tw”를 검색창에 입력하면 빈도 테이블에 기록된 수치를 사용하여 top 5를 계산한다.

상세 설계

- 트라이(trie) 자료구조
- 데이터 수집 서비스
- 질의 서비스
- 트라이 연산
- 규모 확장이 가능한 저장소

트라이(trie) 자료구조

- 기본 트라이 + 빈도에 따라 정렬된 결과를 내놓기 위해 노드에 빈도 정보까지 저장.
- p: 접두어의 길이
- n: 트라이 안에 있는 노드 개수
- c: 주어진 노드의 자식 개수
- 가장 많이 사용된 질의어 k개를 구하는 방법
  - 접두어를 표현하는 노드를 찾는다. O(p)
  - 해당 노드부터 시작하는 하위 트리를 탐색하여 모든 유효 노드를 탐색. O(c)
  - 유효 노드를 정렬하여 가장 인기 있는 검색어 k개를 찾는다. O(clogc)
- 최악의 경우 k개 결과를 얻으려고 전체 트라이 다 검색해야 하는 일이 생길 수 있음. 해결할 방법으로는 다음 두 가지가 있다.
  - 접두어의 최대 길이를 제한
  - 각 노드에 인기 검색어를 캐시

query	frequency
tree	10
try	29
true	35
toy	14
wish	25
win	50

표 13-2

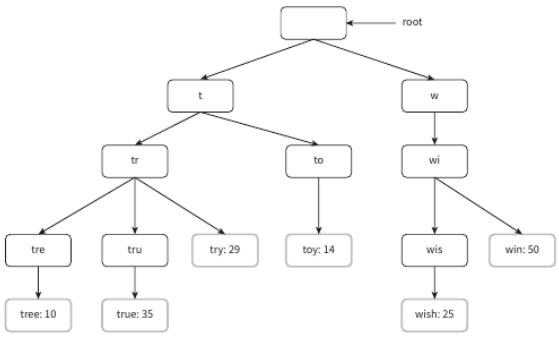

그림 13-6

접두어의 최대 길이를 제한

p값은 작은 정숫값(50)이라고 가정해도 안전하다. 접두어 노드를 찾는 단계에서 시간 복잡도는 O(1)로 바뀔 것이다.

각 노드에 인기 검색어를 캐시

각 노드에 k개의 인기 검색어를 저장해두면 전체 트라이를 검색하는 일을 방지할 수 있다. 5~10개 정도의 자동완성 제안을 표시하면 충분하므로 k는 작은 값이다.

캐시하면 질의 복잡도는 낮출 수 있지만 각 노드의 공간이 많이 필요하게 된다. 그러나 빠른 응답속도가 아주 중요할 때는 이 정도 저장공간을 희생할 만한 가치는 있다.

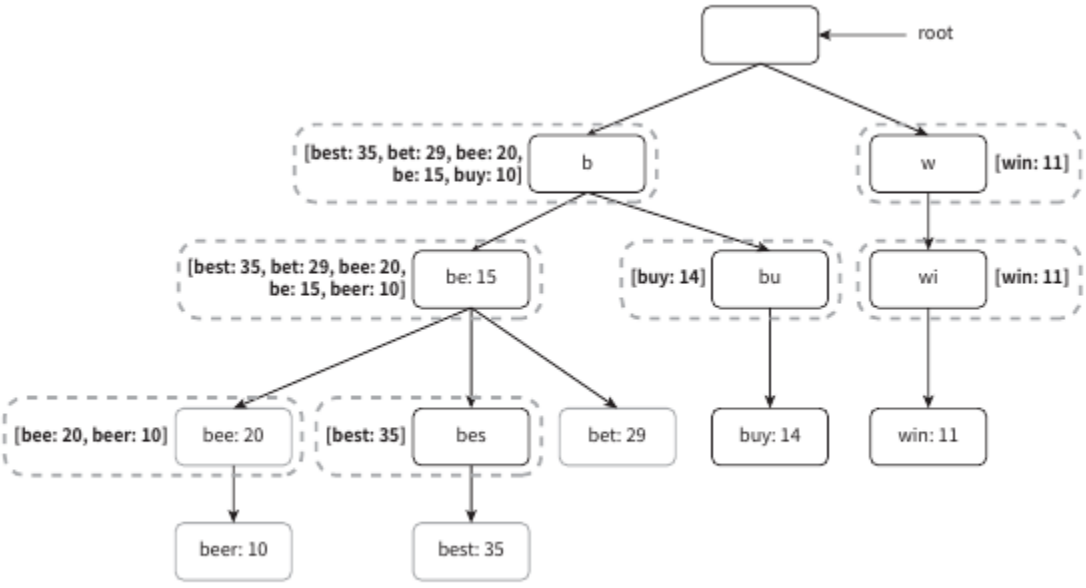

그림 13-8

데이터 수집 서비스

- 매일 수천만 건의 질의가 입력될 때마다 트라이를 갱신하면 질의 서비스는 심각하게 느려질 것이다.
- 일단 트라이가 만들어지고 나면 인기 검색어는 그다지 자주 바뀌지 않을 것이다. 그러니 트라이를 자주 갱신할 필요가 없다.
    - 트위터라면 검색어를 항상 신선하게 유지해야 하지만,
    - 구글 이라면 자주 바꿔줄 이유는 없을 것이다. 트라이를 만드는 데 쓰는 데이터는 보통 데이터 분석 서비스나 로깅 서비스로부터 올 것이고, 데이터 수집 서비스의 토대는 바뀌지 않을 것이기 때문이다.

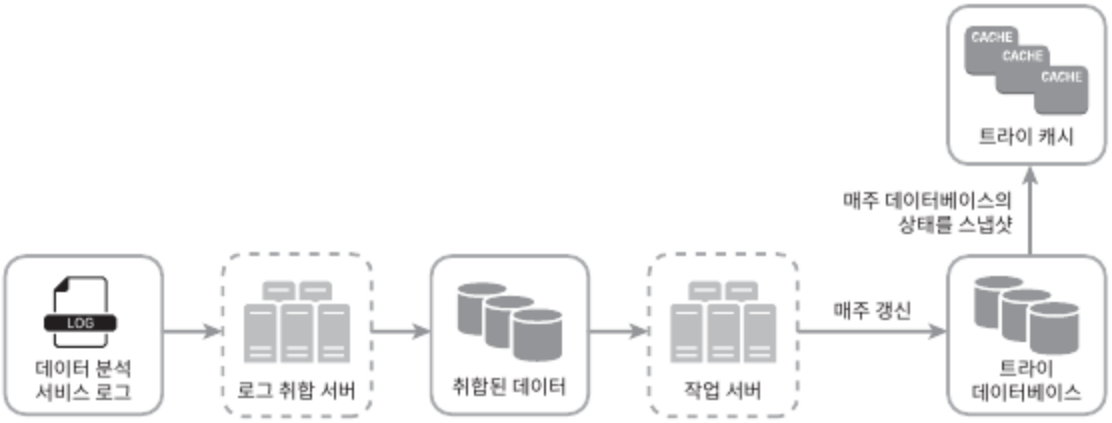

그림 13-9

- 데이터 분석 서비스 로그
    - 검색창에 입력된 질의에 관한 원본 데이터가 보관됨
    - 로그 데이터에는 인덱스를 걸지 않는다.

query	time
tree	2019-10-01 22:01:01
try	2019-10-01 22:01:05
tree	2019-10-01 22:01:30
toy	2019-10-01 22:02:22
tree	2019-10-02 22:02:42
try	2019-10-03 22:03:03

표 13-3

- 로그 취합 서버
  - 데이터 분석 서비스로부터 나오는 로그를 취합하여 시스템이 쉽게 소비할 수 있도록 한다.
  - 데이터 취합 방식은 서비스에 따라 다르다.
  - 실시간 서비스라면 취합 주기를 짧게, 대부분의 경우는 일주일에 한 번 정도로 로그를 취합.
- 취합된 데이터
  - 매주 취합한 데이터. 예시

query	time	frequency
tree	2019-10-01	12000
tree	2019-10-08	15000
tree	2019-10-15	9000
toy	2019-10-01	8500
toy	2019-10-08	6256
toy	2019-10-15	8866

표 13-4

- 작업 서버
  - 주기적으로 비동기적 작업을 실행하는 서버 집합.
  - 트라이 자료구조를 만들고 트라이 데이터베이스에 저장하는 역할

- 트라이 캐시
  - 트라이 캐시는 분산 캐시 시스템으로 트라이 데이터를 메모리에 유지하여 읽기 연산 성능을 높이는 구실을 한다.
  - 매주 트라이 데이터베이스의 스냅샷을 떠서 갱신
- 트라이 데이터베이스
  - 지속성 저장소.
    - 문서 저장소(document store):
      - 새 트라이를 매주 만들 것이므로, 주기적으로 트라이를 직렬화하여 데이터베이스에 저장할 수 있다.
      - 몽고디비같은 문서 저장소를 활용하면 이런 데이터를 편리하게 저장할 수 있다.
    - 키-값 저장소: 트라이는 아래 로직을 따르면 해시 테이블 형태로 변환 가능하다.
      - 트라이에 보관된 모든 접두어를 해시 테이블 키로 변환
      - 각 트라이 노드에 보관된 모든 데이터를 해시 테이블 값으로 변환

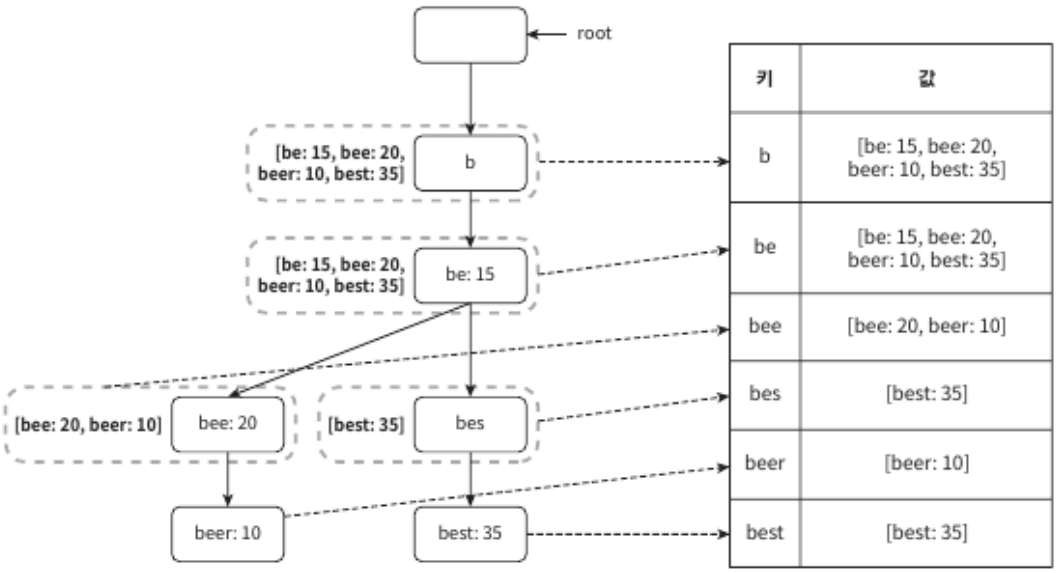

그림 13-10

질의 서비스

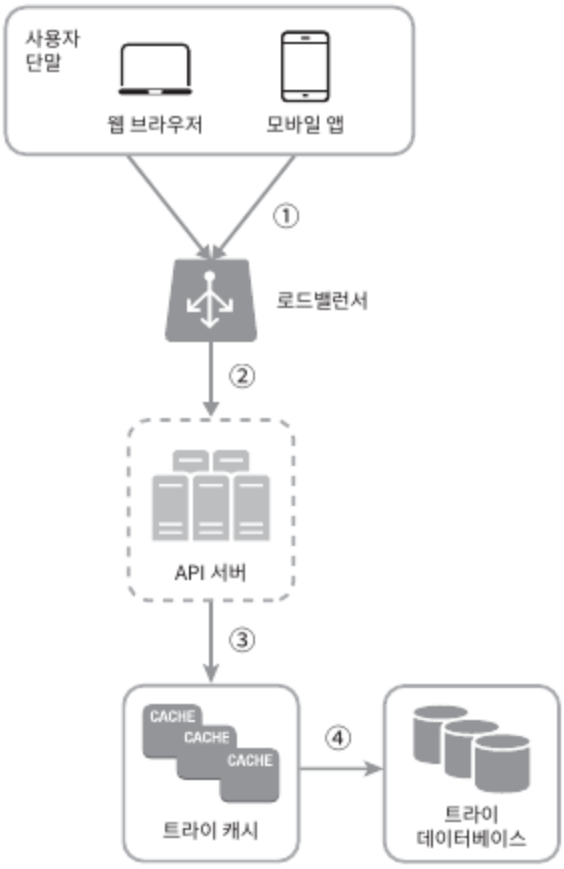

그림 13-11

트라이 연산

검색어 자동완성 시스템의 핵심 컴포넌트.

- 트라이 생성
  - 작업 서버가 담당하며, 데이터 분석 서비스의 로그나 데이터베이스로부터 취합된 데이터를 이용한다.
- 트라이 갱신
  1. 매주 한 번 갱신
     a. 새로운 트라이를 만든 다음에 기존 트라이를 대체한다.
  2. 트라이의 각 노드를 개별적으로 갱신
     a. 성능이 좋지 않음. 갱신할 때 모든 상위노드도 갱신해야 하는데, 상위노드에 인기 검색어 질의 결과가 보관되기 때문이다.

- 검색어 삭제
  - 여러 가지로 위험한 질의어를 자동완성 결과에서 제거해야 한다. 트라이 캐시 앞에 필터 계층(filter layer)을 두고 부적절한 질의어가 반환되지 않도록 한다.
  - 필터 계층을 두면 필터 규칙에 따라 검색 결과를 자유롭게 변경할 수 있다는 장점이 있다.
  - 물리적인 삭제는 다음번 업데이트 사이클에 비동기적으로 진행하면 된다.

그림 13-14

규모 확장이 가능한 저장소

트라이의 크기가 한 서버에 넣기에 너무 큰 경우에 대응해보자.

1. 영어만 지원하면 되기 때문에, 간단하게 첫 글자 기준으로 샤딩.
   a. 세 대 서버라면 a~i까지는 1번 서버, j~r까지는 2번, 나머지는 3번 서버에 저장
   b. 사용 가능한 서버는 최대 26대로 제한됨.
   c. 이 이상으로 늘리려면 샤딩을 계층적으로 해야 함.
      i. a로 시작하는 검색어를 네 대로 나눠 보관 → aa~ag는 1번, ah~an은 2번 ao~au는 3번, 나머지는 4번 서버에 보관.
   d. 그럴싸해보이겠지만 c로 시작하는 단어가 x로 시작하는 단어보다 많다는 것을 감안하면 그렇지 않다. 데이터를 각 서버에 균등하게 배분하기가 불가능함.
2. 과거 질의 데이터의 패턴을 분석하여 샤딩한다.
   a. 검색어 대응 샤드 관리자(shard map manager)는 어떤 검색어가 어느 저장소 서버에 저장되는지에 대한 정보를 관리한다.
   b. s로 시작하는 검색어의 양이 u,v,w,x,y,z로 시작하는 검색어를 전부 합친 것과 비슷하다면, s에 대한 샤드 하나와 u~z까지의 검색어를 위한 샤드 하나를 둔다.

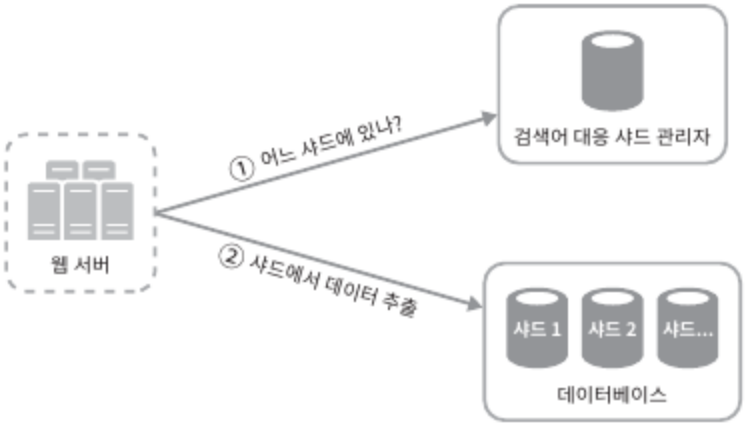

그림 13-15

마무리

- 다국어 지원
  - 트라이에 유니코드 데이터를 저장해야 한다.
  - unicode: 고금을 막론하고 세상에 존재하는 모든 문자 체계를 지원하는 표준 인코딩 시스템
- 국가별 검색어 순위
  - 국가별로 다른 트라이 사용. CDN에 저장하여 응답속도를 높이는 방법도 고려.
- 실시간 검색어 추이 반영
  - 현 설계안은 지원하기에 적합하지 않음.
    - 작업 서버가 매주 한 번만 일하니 시의 적절하게 트라이를 갱신할 수 없다.
    - 트라이 구성시간이 너무 많이 소요된다.
  - 샤딩을 통해 작업 대상 데이터의 양을 줄인다.
  - 순위 모델(ranking model)을 바꾸어 최근 검색어에 보다 높은 가중치를 준다.
  - 데이터가 스트림 형태로 올 수 있다는 점을 고려해야 한다. → 데이터가 지속적으로 생성된다는 뜻.
    - 아파치 하둡 맵리듀스, 아파치 스파크 스트리밍, 아파치 카프카