13장 검색어 자동 완성 시스템

1단계 요구사항

- 자동완성 검색어 : 첫 부분으로 한정
- 개수 : 5개
- 기준 : 인기 순위
- 맞춤법검사 : x
- 언어 : 영어 하지만 된다면 다국어
- 특수문자 : x
- 얼마나 많은 사용자 : DAU 기준 1천만

성능 : < 100ms

연관성 : 입력한 연관된 것

정렬 : 인기도 기반, 순위 모델에 의해 정렬되어야 함

규모 확장성 : 많은 트래픽 감당 가능

고가용성 : 보장되어야 함

개략적인 규모 추정

- DAU : 10million
- 평균 검색 수 : 10
- 질의 시 평균 바이트 : 20 byte (단어 * 크기 = 4 * 5)
- 초당 질의 = 10 million * 10 * 20(type)byte * / 24시간 / 3600 sec = 24000 QPS
- Peek : 48000 QPS
- 신규 검색어 : 20% ⇒ 20byte * 10 * 10000000 * 0.2 ⇒ 0.4GB 신규 검색 데이터

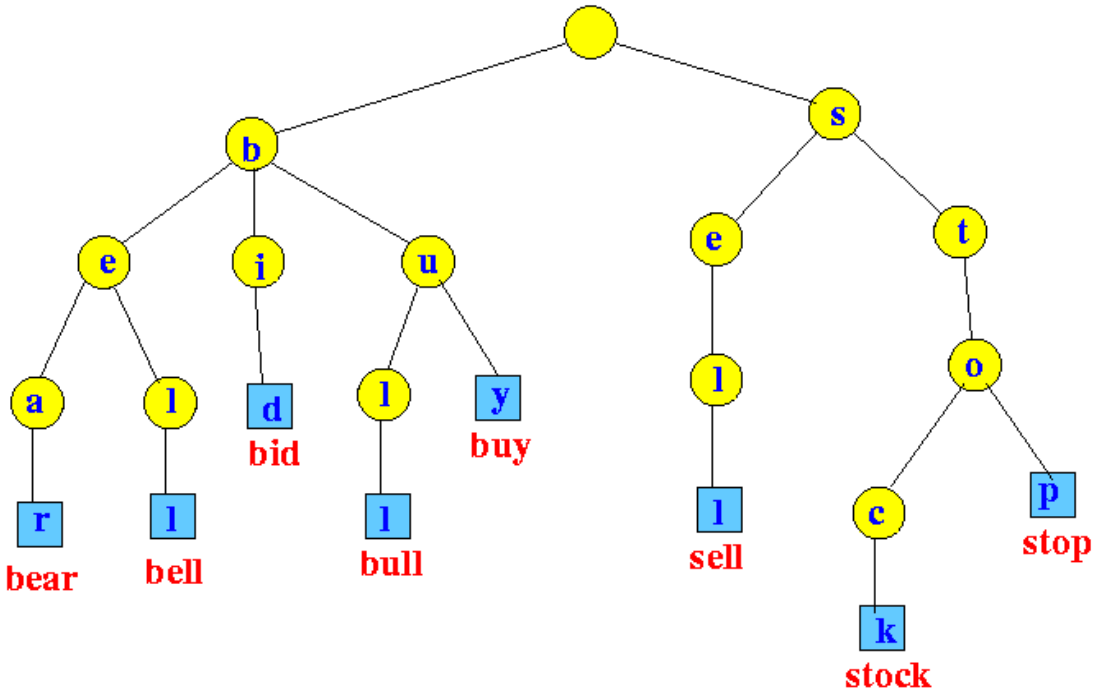
2단계 - 개략적인 설계안 제시 및 동의 구하기

시스템

1. 데이터 수집 시스템 : 입력한 질의를 실시간으로 수집하는 시스템
    a. 빈도 테이블을 이용해서 어떤 질의문이 얼만큼 질의되었는지 카운팅한다
2. 질의 서비스 : 주어진 질의에 다섯 개의 인기 검색어를 정렬해 내놓는 서비스

3단계 - 상세 설계

- **TRIE 자료구조 (Prefix Tree)**



    - 각 노드는 글자 하나를 가지고 있으며 하위 노드는 26개를 가진다
    - 루트 노드부터 리프 노드까지의 경로가 하나의 문자열이다.
    - 인기 검색어를 찾는 방법은
        - 입력된 접두어 하위의 sub tree를 순회하여 전체 문자열을 추리고
        - 추린 문자열에서 빈도수를 기반하여 정렬한 후
        - 상위 N개를 가져오는 방식이다.

        - N + NLogN + 5 이므로 결과적으로 O(NlogN)이다.
    - 하지만 이러한 방식은 최악의 경우 풀서치를 하게 된다는 단점이 있다. 이러한 방법은 아래 두가지 방법으로 해소할 수 있다.
        - 접두어 최대 길이를 제한 → 접두어를 찾는 과정을 상수시간으로 줄일 수 있다.
        - 각 노드에 인기 검색어를 캐시 → 각 노드에 하위 노드 중 인기있는 노드를 캐시하면 상수시간으로 줄일 수 있다.
        - 두가지 방법을 조합하면 전체 검색을 방지하고 상수시간 내에 인기 검색어를 제공할 수 있다.
- **데이터 수집 서비스**
    - 실시간으로 데이터를 수정하는 것은 실용적이지 못하다. 인기 검색어는 자주 바뀌지 않기 때문이다.
    - 개선된 아키텍처 : [데이터 분석 서비스 로그] → [로그 취합 서버] → [취합 데이터] → [작업 서버] → 트라이 데이터베이스] → [트라이 캐시]
        - 데이터 분석 서비스 로그 : 질의 로그에 대한 원본 데이터 시계열
        - 로그 취합 서버 : 로그 데이터를 집계하여 쉽게 소비할 수 있도록 해야함. 일주일 주기로 한다고 가정
        - 취합된 데이터 : 취합 서버의 결과물
        - 작업 서버 : 비동기적으로 작업을 실행하는 서버의 집합 → 트라이 데이터 베이스에 저장하는 역할
        - 트라이 캐시 : 트라이 데이터를 메모리에 유지하여 읽기 연산 성능을 높힘
        - 트라이 데이터베이스 : 지속성 저장소
            - 문서 저장소 : 새 트라이를 매주 만들 것이므로 트라이를 직렬화 해서 저장
            - 키 값 저장소 : 접두사를 키로 변환하여 저장함 인기 있는 상수 개의 결과만 값으로 저장 (리얼 캐시 서비스인 것 같다)
- **질의 서비스**
    - 아키텍처 : 클라이언트 → 로드밸런서 → API Servers → TRIE Caches → TRIE Database
    - 특별히 추가적으로 설명할만한 내용이 없다.
    - 최적화 방안 : ajax, browser caching, data sampling(N개 중 몇개만 샘플링하여 로깅)

트라이 연산?

- 트라이 갱신
  - 매주 한번 갱신 - 새로운 트라이를 만들고 대체하는 방식
  - 각 노드를 개별적 갱신 → 성능이 좋지 않다. 왜냐하면 개별 단어별로 갱신할 때에 상위 노드의 데이터도 모두 변경해야하기 때문이다. 데이터가 적을 때에는 충분히 유효할 수 있으나 데이터가 많고 깊은데, 인기있는 검색어 같은 경우에는 너무 많은 노드를 수정해야할 수 있다. (그래봐야 최대 검색어수준 아닌가? 잘 이해안감)
    - 리얼타임 시스템에서는 조금 유의미할지도
- 검색어 삭제
  - 혐오적이거나 폭력적, 성적 노골적인 위험한 질의어는 필터 계층을 통하여 결과에서 제거해야 한다.
  - 필터계층에서 실시간으로 제거하고 다음번 배치를 진행할 때 영구적으로 제거한다.

저장소 규모 확장?

- 검색어를 기준으로 테이블을 샤딩한다. 문자의 범위를 기준으로 데이터베이스를 샤딩한다.
- 하나의 문자에 너무 많은 데이터가 있다면 계층적으로 데이터를 샤딩하는 방법도 있다. AA → 1Node, AB → 2Node
- 하지만 N분할 하는 샤딩은 사실상 균등하게 데이터가 분할되지 않으므로, 검색어 샤드 관리자를 통해 데이터에 사이즈에 적절하게 끔 분할한다. (예: A* → 1번 서버, B~D → 2번 서버)

마무리

1. 다국어 지언 시스템 확장
2. 국가별 인기 검색어 순위가 다른 경우
3. 실시간 검색어 추이 반영
   a. 샤딩을 통해 작업 대상 데이터 양을 줄이고
   b. 순위 모델을 사용하여 최근 검색어에 보다 높은 가중치를 주고
   c. 데이터 스트림 처리에 특화된 컴포넌트를 이용하여 집계한다.

d. Sliding window 방식으로 특정 범위를 매 5분 마다 분석하여 인기 검색어를 만들 수도 있을 것 같다.