웹 크롤러 설계

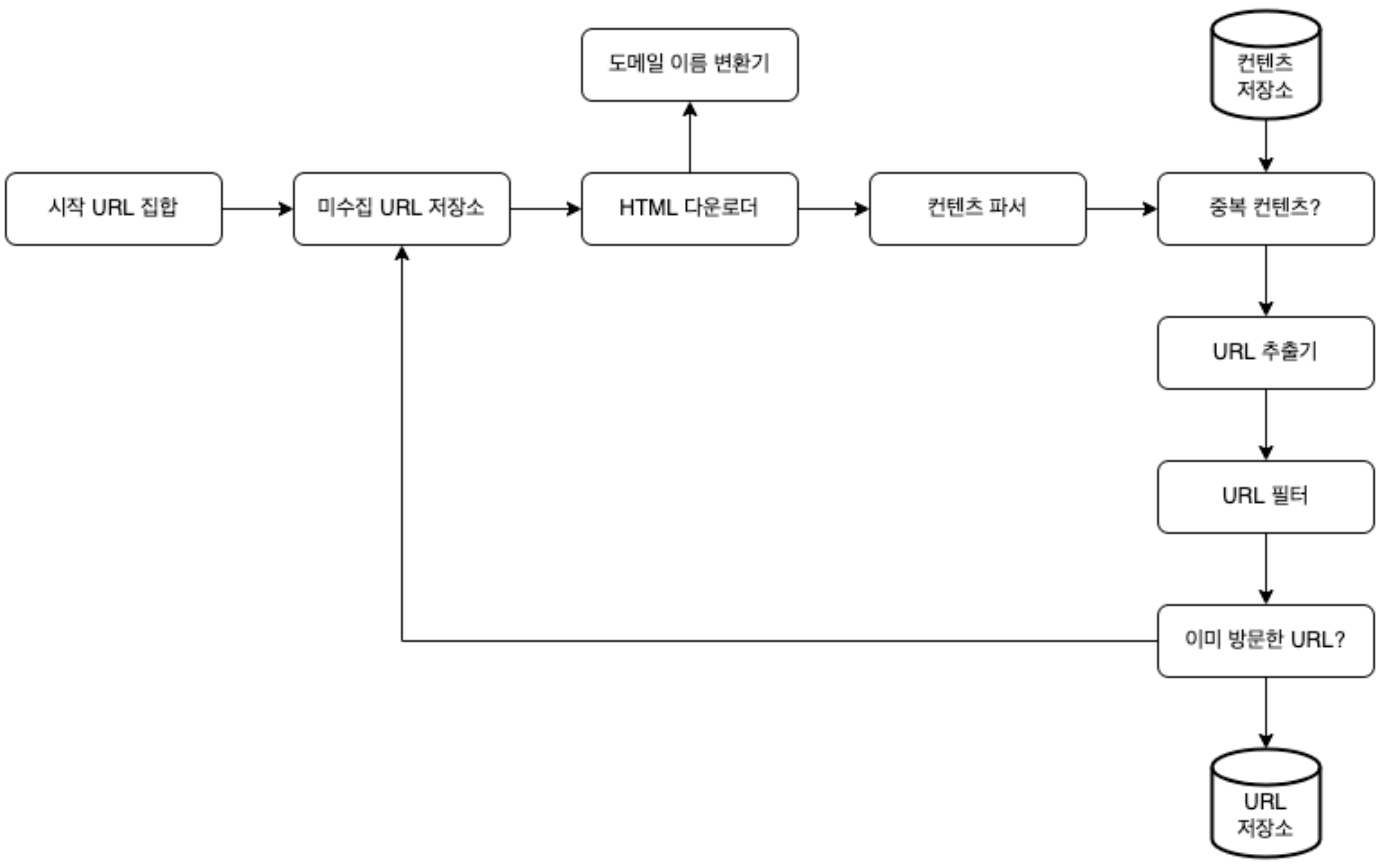
웹 크롤러의 사용처

- 검색 엔진 인덱싱
- 웹 아카이빙
- 웹 마이닝
- 웹 모니터링

문제 이해 및 설계 범위 확정

- 매달 10억개의 웹페이지 다운로드
- QPS = 10억 / 30일 / 24시간 / 3600초 = 400페이지/초
- maximum QPS = QPS *2 = 800페이지 / 초
- 웹페이지의 크기 = 500KB (평균)
- 10억 * 500KB = 500TB / 월
- 5년간 30PB

개략적 설계안 제시 및 동의 구하기

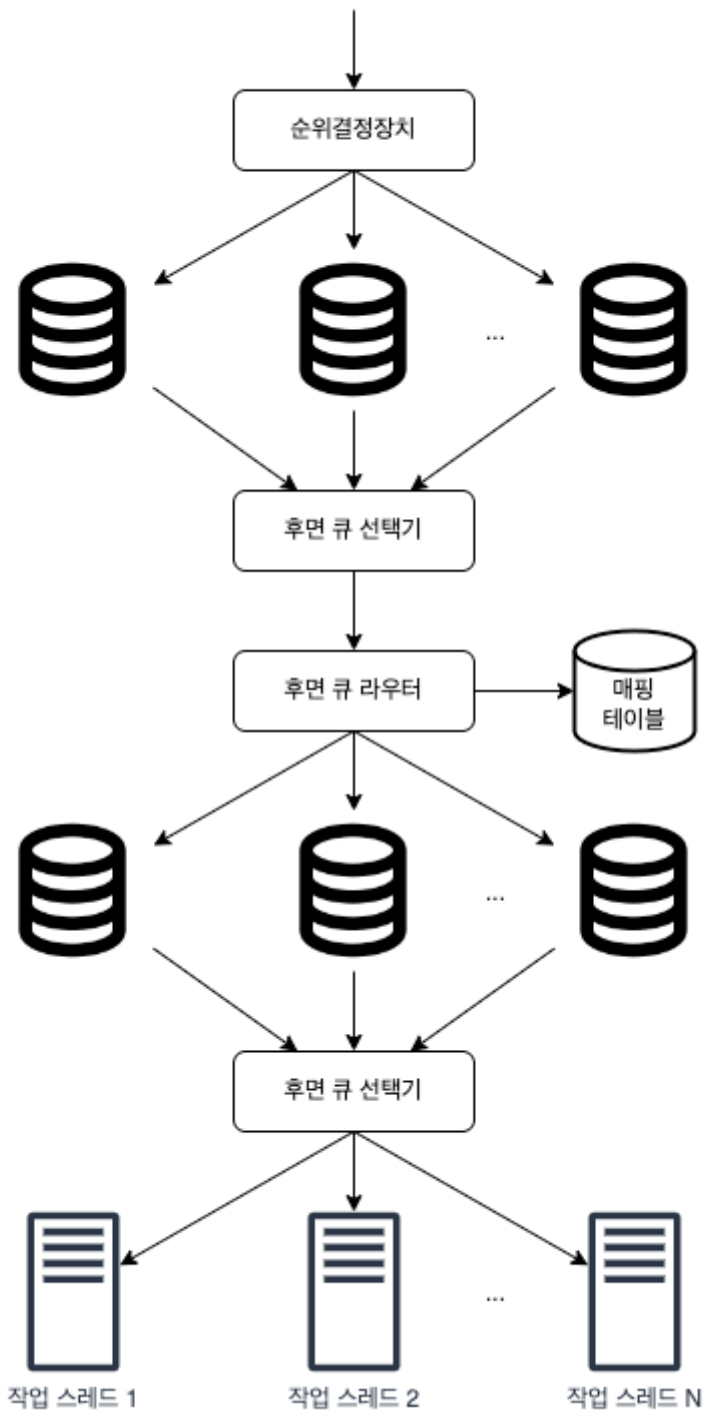


- 시작 URL 집합 : 웹 크롤러가 크롤링을 시작하는 출발점
- 미수집 URL 저장소 : 다운로드 할 예정인 URL, FIFO queue.
- HTML 다운로더 : HTML 다운로더는 인터넷에서 웹 페이지를 다운로드하는 컴포넌트
- 도메인 이름 변환기 : 웹 페이지를 다운받으려면 URL을 IP 주소로 변환하는 절차가 필요
- 컨텐츠 파서 : 파싱과 검증 절차를 함께 수행
- 중복컨텐츠 : 웹에 공개된 연구 결과에 따르면, 29% 가량의 웹 페이지 컨텐츠는 중복
    - 이미 시스템에 저장되어 있는지 알아내는 쉽고 효과적인 방법은 웹 페이지의 해시 값을 비교하는 것.
- 콘텐츠 저장소 : 데이터 양이 너무 많으므로 대부분의 콘텐츠는 디스크에 저장

    - 인기 있는 콘텐츠는 메모리에 두어 접근 지연시간을 줄일 수 있음
- URL 추출기 : URL 추출기는 HTML 페이지를 파싱하여 링크들을 골라내는 역할
- URL 필터 : URL 필터는 특정한 콘텐츠 타입이나 파일 확장자를 갖는 URL, 접속 시 오류가 발생하는 URL
    - 접근 제외 목록에 포함된 URL 등을 크롤링 대상에서 배제하는 역할
- 이미 방문한 URL : 블룸 필터나 해시 테이블을 적절히 사용하기
- URL 저장소 : 이미 방문한 URL을 보관하는 저장소

상세 설계

- DFS vs BFS
    - 웹은 directed graph
        - 페이지는 노드, URL은 엣지
    - 크롤링에서 DFS를 쓸 경우, 어느 정도로 깊숙이 가게 될지 가늠이 되지 않음
        - 따라서 웹 크롤러는 보통 BFS를 사용
    - 그러나 구현법에는 두 가지 문제점이 있음
        - 한 페이지에서 나오는 링크의 상당수는 같은 서버로 되돌아 감
        - 표준적 BFS 알고리즘은 우선순위가 없음
- 미수집 URL 저장소
    - 미수집 URL을 통해 위의 두가지 문제를 해결할 수 있음
- 예의
    - 웹 크롤러는 수집 대상 서버로 짧은 시간 안에 많은 요청을 보내는 것을 삼가야 함
- 우선순위
    - URL 마다 페이지 랭킹을 매길 수 있음



- 전면 큐
  - 우선순위 결정 과정을 처리
- 후면 큐
  - 크롤러가 예의 바르게 보증하도록 보증

- 신선도
  - 웹 페이지는 수시로 추가되고, 삭제되고, 변경되므로 데이터의 신선함을 유지하기 위해서는 주기적으로 업데이트 해줘야 함
  - 이를 최적화하는 전략은 다음과 같음.
    - 웹 페이지의 변경 이력(update history) 활용
    - 우선순위를 활용하여, 중요한 페이지는 좀 더 자주 재수집

- 미수집 URL 저장소를 위한 지속성 저장장치
  - URL을 모두 메모리에 보관하는 것은 적합하지 않음

- HTML 다운로더
  - Robots.txt
    - 웹사이트가 크롤러와 소통하는 표준적인 방법
  - 성능 최적화

        - 분산 크롤링
            - 성능을 높이기 위해 크롤링 작업을 여러 서버에 분산
    - 도메인 이름 변환 결과 캐시
        - DNS Resolver는 크롤러 성능의 병목 중 하나
        - DNS 조회 결과로 얻어진 도메인 이름과 IP 주소 사이의 관계를 캐시에 보관해 놓고 크론 잡 등을 돌려 주기적으로 갱신하도록 하면 성능을 효과적으로 높일 수 있음
    - 지역성
        - 크롤링 작업을 수행하는 서버를 지역별로 분산하는 방법
    - 짧은 타임아웃
        - 어떤 웹 서버는 응답이 느리거나 아예 응답하지 않으므로, 타임아웃 설정 필요

- 안정성
    - 안정 해시(consistent hashing) : 다운로드 서버들에 부하를 분산할 때 적용 가능한 기술
    - 크롤링 상태 및 수집 데이터 저장 : 장애가 발생한 경우에도 쉽게 복구할 수 있도록 크롤링 상태와 수집된 데이터를 지속적 저장장치에 기록해 두는 것이 바람직함
    - 예외 처리(exception handling) : 대규모 시스템에서 에러(error)는 불가피할 뿐 아니라 흔하게 벌어지는 일
    - 데이터 검증(data validation) : 시스템 오류를 방지하기 위한 중요 수단 가운데 하나

- 문제 있는 콘텐츠 감지 및 회피 전략
    - 중복 콘텐츠
        - 웹 컨텐츠의 30% 가량은 중복.
    - 거미 덫
        - 크롤러를 무한 루프에 빠뜨리도록 설계한 웹 페이지
        - 최대 길이를 제한함으로 회피할 수 있음
    - 데이터 노이즈
        - 어떤 콘텐츠는 가치가 없음

마무리

- 서버 측 랜더링(SSR): 많은 사이트가 자바 스크립트, AJAX 등의 기술을 사용
- 원치 않는 페이지 필터링: 저장 공간 등 크롤링에 소요되는 자원은 유한하기 때문에 스팸방지 컴포넌트로 두어 품질이 조악하거나 스팸 성인 페이지를 걸러내도록 설정
- 데이터베이스 다중화 및 샤딩: 다중화나 샤딩 같은 기법 적용시, 데이터 계층의 가용성,규모 확장성, 안정성이 향상.
- 수평적 규모 확장성: 서버가 상태정보를 유지하지 않도록 하는 것, 즉 무상태(stateless)서버로.
- 가용성, 일관성, 안정성: 필수적으로 고려할 사항
- 데이터 분석 솔루션(analytics): 데이터를 수집하고 분석하는 것은 어느 시스템에게나 중요