9장 — 웹 크롤러 설계

웹 크롤러의 다양한 사용처

- 검색 엔진 인덱싱
- 웹 아카이빙
- 웹 마이닝
- 웹 모니터링

1단계) 문제 이해 및 설계 범위 확정

아래 속성들을 고려하여 요구사항을 명확히 하자

- 규모 확장성 — 현대 웹은 매우 거대 (수십억 개의 페이지 존재)
- 안정성 — 웹은 함정이 많다. (악성 코드, 잘못 작성된 HTML)
- 예절 — 짧은 시간동안 너무 많은 요청을 웹 사이트에 보내면 안된다.
- 확장성 — 새로운 형태의 콘텐츠 지원하기 쉬워야 한다.

개략적 규모 추정

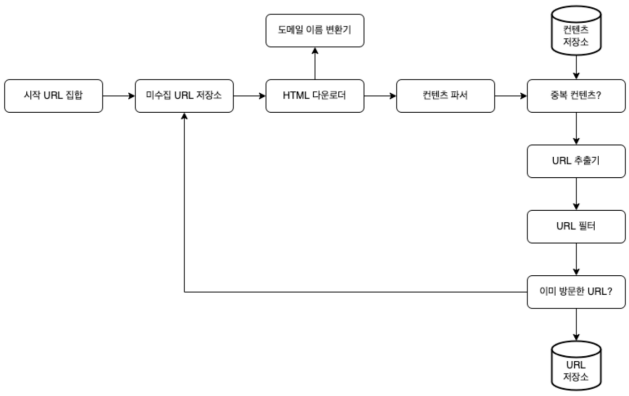
- 매달 10억개의 웹페이지 다운로드
- QPS = 10억 / 30일 / 24시간 / 3600초 = 400페이지/초
- maximum QPS = QPS *2 = 800페이지 / 초
- 웹페이지의 크기 = 500KB (평균)
- 10억 * 500KB = 500TB / 월
- 500TB / 월 * 12개월 * 5년 ⇒ 5년간 30PB

2단계) 개략적 설계안 제시 및 동의 구하기

1. 시작 URL 집합: 크롤링을 시작하
   는 출발점

    a. 다양한 링크를 탐색할 수 있도록 지역적 특생, 주제별로 시작 가능

2. 미수집 URL 저장소: 다운로드 해야하는 URL 저장
3. HTML 다운로더: 저장소에서 다운로드할 URL을 받아 웹 페이지 다운로드
4. 도메인 이름 변환기: URL을 ip로 변환
5. 콘텐츠 파서: parsing, validtaion 절차를 거쳐, 이상한 페이지 거르기
6. 중복 콘텐츠?: 웹의 29% 가량은 콘텐츠 중복이다.
    a. 중복을 확인하는 방법: 직접 비교는 10억 개의 문서 비교 사실상 불가능
    → 해시를 통해서 확인
7. 콘텐츠 저장소: HTML 문서 보관
→ 메모리 VS 디스크
8. URL 추출기: HTML을 파싱하여 링크들을 골라내는 역할
9. URL 필터: 특정 콘텐츠 타입, 특정 파일 확장자를 갖는 URL, 접속 시 오류가 발생하는 URL을 크롤링 대상에서 배제하는 역할
10. 이미 방문한 URL?: 이미 방문한 URL인지 확인
    a. bloom filter,해시 테이블을 주로 사용
11. URL 저장소: 이미 방문한 URL을 보관하는 저장소



3단계) 상세 설계

중요한 컴포넌트를 살펴보자

- `DFS` VS `BFS`
- HTML 다운로더
- 안정성 확보 전략
- 확장성 확보 전략
- 문제 있는 콘텐츠 감지 및 회피 전략

`DFS` VS `BFS`

DFS를 사용할 때, 그래프의 크기가 크다면 어느 정도로 깊숙이 가게 될지 가늠하기가 어려워진다.

→ 대부분 웹 크롤러는 BFS를 사용한다. FIFO 큐를 사용해 탐색할 URL을 보관한다.

문제점

- 한 페이지에서 나오는 링크는 상당수 같은 서버로 되돌아간다.
- 표준적 BFS는 URL간에 우선순위를 두지 않는다.

미수집 URL 저장소

- 미수집 URL을 통해 위의 두가지 문제를 해결할 수 있다.

예의

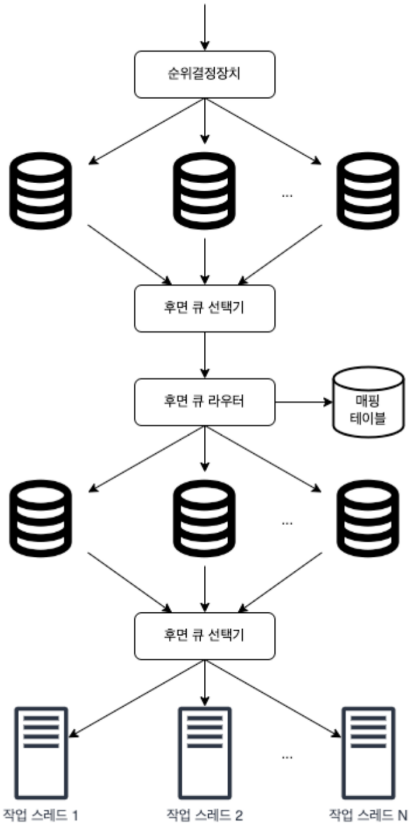
- 짧은 시간동안 너무 많은 요청을 보내는 것은 무례한 일이다. → 사이트 마비, DoS 공격으로 간주될 수 도 있다.
- 동일 웹사이트에 대해서는 한 번에 한 페이지만 요청해야한다.

우선순위

- 페이지마다 중요도가 다르기 때문에, URL 의 우선순위를 두어 높은 우선순위 우선 탐색한다.

전면 큐: 우선순위 결정 과정을 처리한다.

후면 큐: 크롤러가 예의 바르게 동작하도록 보증한다



신선도

- 웹 페이지는 수시로 추가, 삭제, 변경된다.
  - 데이터의 신선함을 유지하기 위해서는 이미 다운로드 한 페이지라고 하더라도 주기적으로 재수집을 해야한다.
- 모든 URL을 재수집하는 것은 너무 많은 자원과 시간이 필요
  - 웹 페이지의 변경 이력을 활용하거나, 우선순위를 활용해 중요한 페이지를 더 자주 재수집 하는 방법이 있다.

HTML 다운로더

- Robots.txt
  - 웹 사이트가 크롤러와 소통하는 표준적인 방법

- 성능 최적화
  - 분산 크롤링: 크롤링 작업을 여러 서버에 분산하는 방법
  - 도메인 이름 변환 결과 캐시: DNS를 통해 얻어진 도메인 이름과 ip 주소 사이의 관계를 캐시에 보관해 놓고 크론 잡으로 주기적으로 갱신하도록 한다.
  - 지역성: 크롤링 작업을 수행하는 서버를 지역별로 분산
  - 짧은 타임아웃: 최대 어느정도 응답을 기다릴 것인지 미리 설정한다.
- 안정성
  - 안정해시: 서버들의 부하 분산, 쉽게 서버 추가 삭제 가능한 안정 해시 방법 사용 가능
  - 크롤링 상태 및 수집 데이터 저장: 장애가 발생했을 때 쉽게 복구 가능하도록 하기 위해, 지속적으로 저장장치에 기록
  - 예외 처리: 예외가 발생해도 전체 시스템이 중단되지 않도록 처리
  - 데이터 검증: 시스템 오류 방지하기 위해 중요 수단 가운데 하나
- 확장성
  - 새로운 콘텐츠를 지원할 수 있도록 쉽게 변화 가능한 구조 필요
- 문제 있는 콘텐츠 감지 및 회피
  - 중복 콘텐츠: 해시, 체크섬을 통해 중복 콘텐츠를 탐지
  - 거미 덫: 무한 루프에 빠뜨리도록 설계된 웹 페이지 → 최대 길이를 제한하면 회피 가능
  - 데이터 노이즈: 가치가 없는 콘텐츠도 많다. 가능하다면 이러한 콘텐츠 제외

4단계) 마무리

- 서버 측 랜더링: 많은 사이트들이 JS, AJAX등을 기술을 사용, 동적으로 링크를 생성하는데, 이 링크들은 크롤러가 발견 할 수 없다.
- 원치 않는 페이지 필터링: 항상 자원은 유한하기 때문에, 스팸 방지 컴포넌트를 두고 품질이 떨어지거나 스팸성인 페이지를 걸러내자
- 데이터베이스 다중화 및 샤딩: 다중화, 샤딩을 통해 데이터 계층의 가용성, 규모 확장성, 안정성이 향상된다.

- 수평적 규모 확장성: 다운로드할 서버가 수백, 수천 대 필요할 수 있다. 이를 위해 수평적 규모 확장성을 달성하기 위해 무상태 서버로 만들어야 한다.
- 가용성, 일관성, 안정성: 대형 시스템의 필수적으로 고려해야하는 사항
- 데이터 분석 솔루션: 시스템을 세밀히 조정하기 위해서는 데이터를 수집하고 분석해야 한다.